University Oldenburg

Wind Physics Measurement Project

Exercise 4 - Lidar measurements

Author:
Jan Kämper
Florian Börgel

Supervisor:
Marijn Floris van Dooren
Julian Hieronimus

July 6, 2016

Contents

Introduction

The fourth exercise deals with Lidar measurements. The exercise itself was separated into three sub tasks. First we received 312 frequency spectra which span a vertical plane in front of the SpinnerLidar. For each spectrum we calculated the peak and the corresponding line-of-sight velocity.

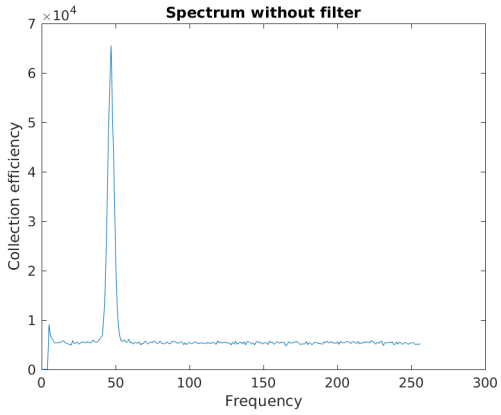
1 Task 1: Spectral analysis: from backscatter spectrum to line-of-sight velocity

1.1 Clean the spectra from their background noise

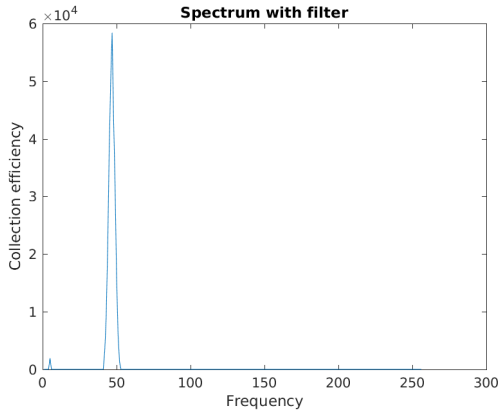
In order to clean the 312 different spectra from their individual background noise we calculated the mean values of each spectra. Keeping in mind that the goal of this exercise is to identify the peak location, we defined 1.5 multiplied with the mean of the spectrum as noise. Therefore we subtracted $1.5 \cdot mean$ and discarded all values below zero. The following code-snippet shows this procedure in Matlab:

```
for pos = 1:312
    spinner_noiseCancelled(pos,:) = spinnerlidar_spectra(pos,:) -1.5*mean(
    spinnerlidar_spectra(pos,:));
    spinner_noiseCancelled(spinner_noiseCancelled < 0) = 0;
end;
```

The following figure shows on the left the raw spectrum and on the right the noise cleaned spectrum. As one can easily see we obtain different values for the collection efficiency, but since it is only important to find the corresponding frequency and subsequently the line-of-sight velocity of the peak, the peak value itself is not relevant for further calculations.



(a) Raw spectrum



(b) Spectrum cleaned from background noise

1.2 Calculate the Doppler frequency that each bin of the spectra corresponds to

We received the spectra as histograms which are separated in 256 bins. The measured bandwith is $25 \cdot 10^6$ Hz. Therefore each bin represents a frequency of

$$frequency = x * \frac{25 \cdot 10^6 Hz}{256}$$

where x is the bin number. With the equations provided during the lecture we were also able to calculate the line-of-sight velocity. The implementation in Matlab is shown in the following.

```
for bin=1:256
    f_d(bin,1) = (bin-1)/bins*bandwith;
    v(bin,1) = f_d(bin,1) *lambda_0;
end;
```

1.3 For each spectrum, define the peak location with the centroid method

The centroid method was introduced during the lecture and is defined as:

$$f_{peak} = \frac{\int f_d \cdot p(f) df}{\int p(f) df}$$

Before we calculated the peak values we normalized our data to get a PDF instead of absolute values. Since we are dealing with discrete values, we summed over all values in order to identify

the peak value. In order to check all 312 f_{peak} values, we first calculated the index of the maximum collection efficiency for each spectrum. The maximum should always represent the peak of the spectrum. Next we determined the corresponding index of f_{peak}. Now we were able to compare the distance of the calculated indices. The results are shown in figure 2.

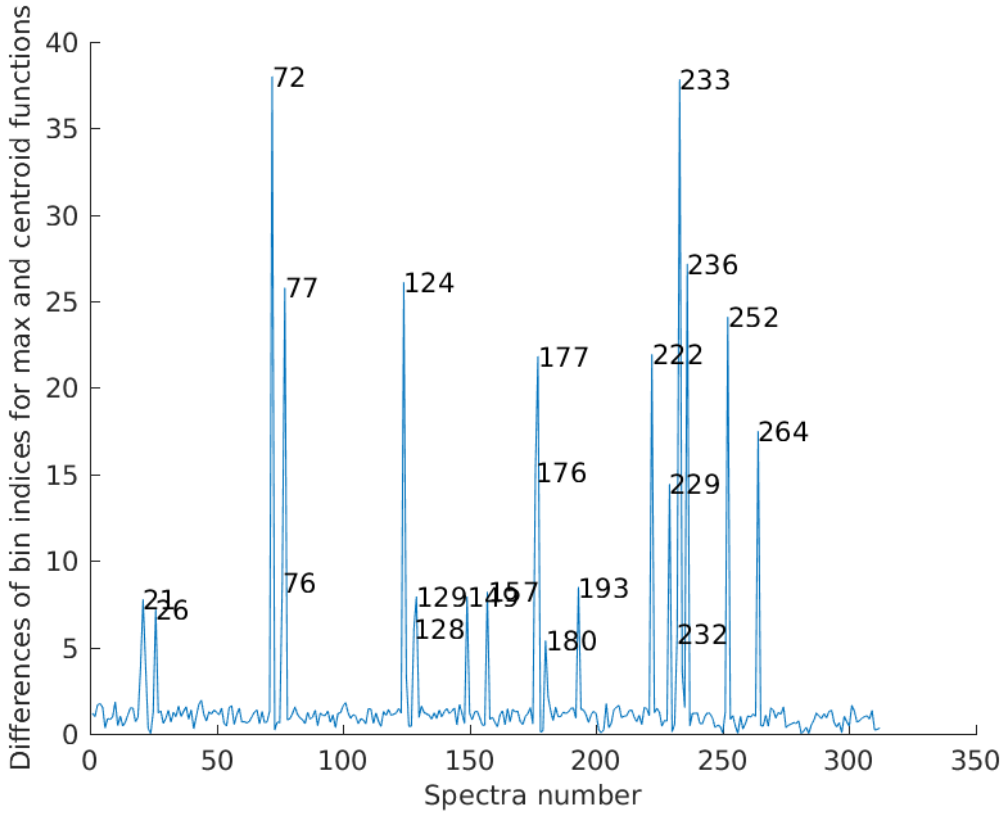


Figure 2: Centroid Errors in bin distance

The figure shows that there are several errors. The centroid method fails for some spectra due to a measurement error. As stated in the exercise sheet the lidar was placed on the nacelle of a wind turbine. Therefore for some measurements the lidar might be blocked by the rotating blades which would result in a second frequency peak. If the spectrum contains two peaks, the centroid method returns the middle of these two peaks and is not applicable for this case. One of those cases is shown in figure 3.

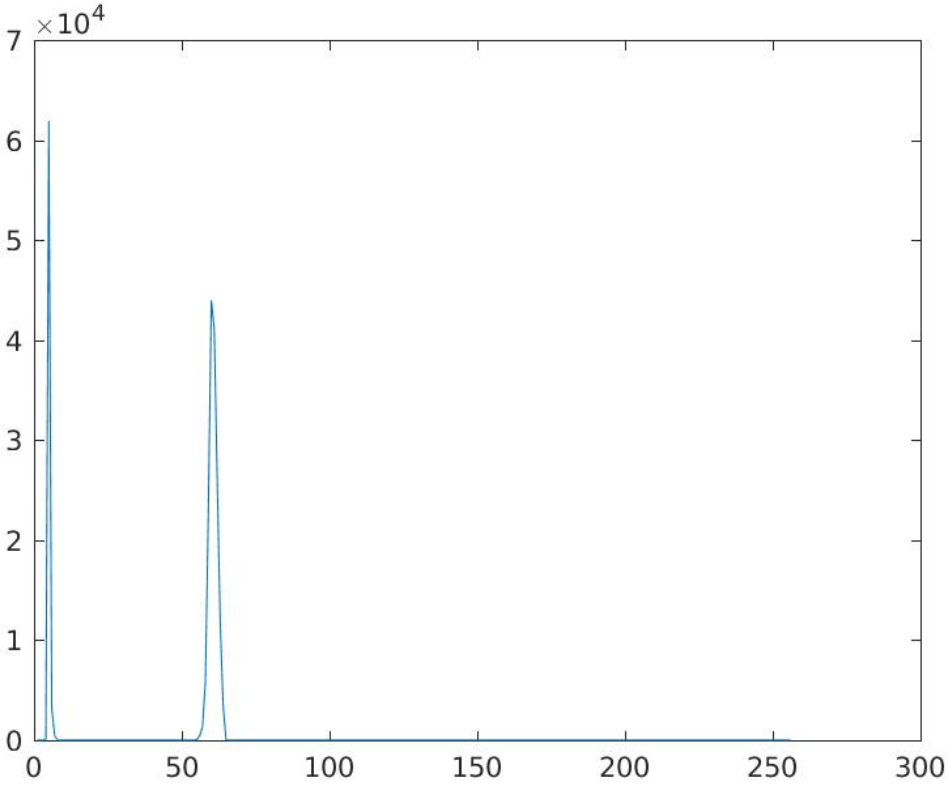


Figure 3: Measurement error in spectrum 72

1.4 Correlate the calculated line-of-sight speeds

In step 4 we were asked to correlate the calculated line-of-sight speeds with the lidar data itself. The correlation method was already provided. Therefore we only needed to calculate the line-of-sight velocity by multiplying f_{peak} with λ_0. The resulting plot is shown in figure 4.

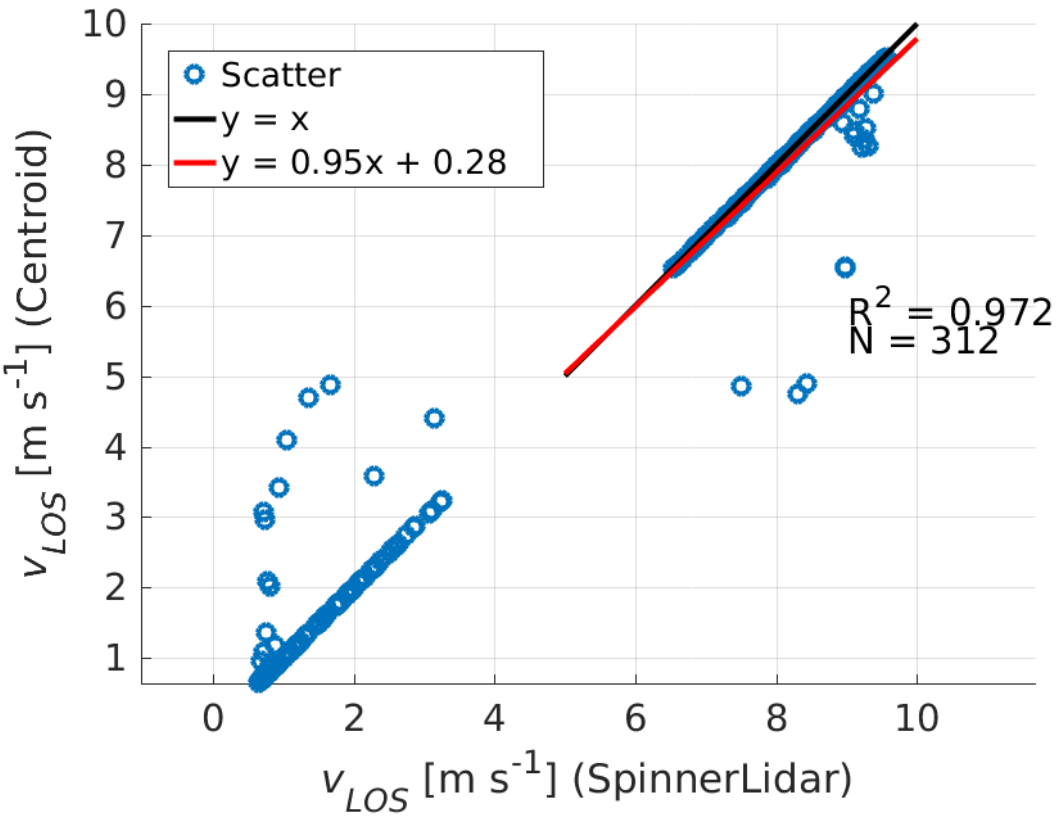


Figure 4: Correlation of the calculated line-of sight-speed and the lidar data

Figure 4 shows a strong correlation of $R^2 = 0.972$. As discussed in section 1.3 there are measurement errors due to the position of the lidar. Some measurements are blocked by the moving blades.

2 VAD Scanning: from line-of-sight velocity to wind factor

2.1 Investigate the LiDAR data

In exercise 2 we were working with a 6 hour dataset of VAD Lidar measurements located on the FINO1 platform. For further comparison we were also provided a 10 minute averaged dataset of FINO 1. The first step was to filter bad measurements. A common procedure is to filter bad Carrier-to-Noise-Ratios (CNR). The procedure is shown in the following code-snippet:

```
% Filter data by CnR, by setting to NaN and later in Step 2 discarding NaN
% values
cnr_ind = find(cnr_VAD <= -20 | cnr_VAD >= -5);
rs_VAD(cnr_ind) = NaN;
az_c_VAD(cnr_ind) = NaN;
```

After filtering the data we were asked to separate the data for the different range gates. We identified 14 different range gates, measuring at $61m, 64m, 67m, 70m, 72m, 75m, 78m, 81m, 84m, 87m, 90m, 93m, 96m$, and $98m$. The last task of 2.1 was to calculate the time for a full 360° scan. The LiDAR has a scanner speed of $25°/s$. Therefore one full scan takes:

$$time\ per\ scan = \frac{360}{25°/s} = 14.4s$$

The given data is measured at a time interval of 0.4 s. A full scan represents $14.4s/0.4s = 36$ intervals.

2.2 Separate the data single 360° scans and carry out a cosine fit

The goal of this section is to compute the horizontal wind speed, the horizontal wind direction and the vertical wind speed. In the last section we calculated that a full scan is represented by 36 intervals. So in the ideal case of a homogeneous atmosphere the line-of-sight velocity should show a sine-like behaviour. There fore we should be able to fit this to a function of type:

$$v_{rad} = b \cdot \cos(\phi - \Theta) + a$$

where

$$v_{hor} = \frac{b}{cos(\theta)} \qquad w = \frac{-a}{\sin(\theta)} \qquad D = b \pm 180$$

To implement the fitting routine in Matlab we first defined the function, depending on the three parameters b, Θ and a and the azimuth angle ϕ. During the lecture the function $lsqcurvefit()$ was recommended as the fitting routine. It should be noted, that the function is not able to deal with NaN values. Therefore NaN values had to be discarded temporally. The following code-snippet shows the Matlab implementation. The fitting was done over all full 360° scans and over all range gates.

```
for j = 1:14 % number of Range Gates
    for i = 1:numberOfScans
        phi = az_c_VAD_filter((i-1)*intervalls_per_scan+18:(i-1)*intervalls_per_scan
    +53,j);
        v_r = rs_VAD_filter((i-1)*intervalls_per_scan+18:(i-1)*intervalls_per_scan
    +53,j);
        nanIndices = isnan(phi) | isnan(v_r);
        phi(nanIndices) = [];
        v_r(nanIndices) = [];
        if length(phi>0)
            fitparam = lsqcurvefit(VADCos, startvalues, phi, v_r);
            savedparam{i,j} = fitparam;
            %calculate horiz and vertical speed
            v_hor = fitparam(1) / cosd(60);
            v_ver = -fitparam(3) / sind(60);
            D = abs(fitparam(1)-180);
            calc_times(i,j) = ts_VAD_filter((i-1)*intervalls_per_scan+18,j);
            calc_vHor(i,j) =  v_hor;
            calc_vVer(i,j) = v_ver;
            calc_D(i,j) = D;
        end
    end
end
```

In order to understand what the fitting routine returns, we also plotted the data and the fitted curve. The results are shown in figure 5.

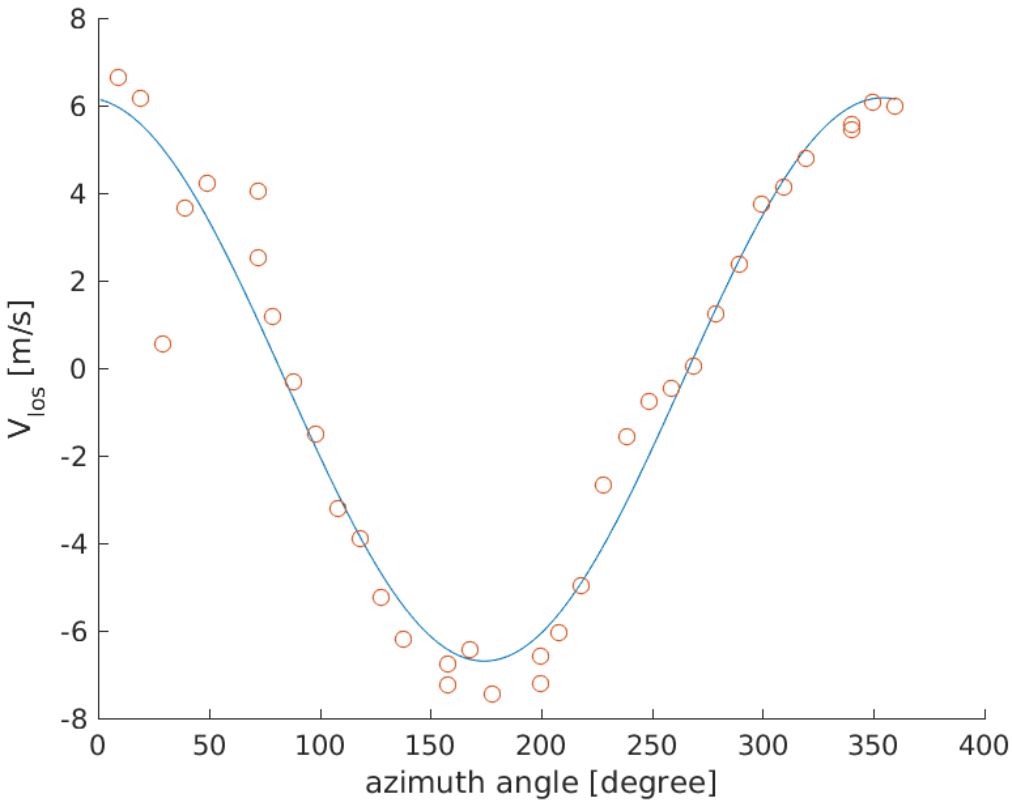


Figure 5: Full 360° for Range Gate = 61 m with fit

2.3 10 minute averages and comparison of vertical wind profiles and wind directions

The FINO 1 data includes wind vanes at heights of *33m, 40m, 50m, 60m, 70m, 80m, 90m* and eight anemometers at heights *33m, 40m, 50m, 60m, 70m, 80m, 90m and 100m.*The measurements are in 10 minute intervals. The LiDAR data was measured at heights *67m, 70m, 72m, 75m, 77m, 80m, 83m* and *85m.* To compare the vertical profiles we had to compute 10 minute averages of the LiDAR wind speeds and wind directions. Since we only had a six hour dataset we searched for the corresponding time stamp in the FINO 1 data. After averaging the LiDAR data and extracting the correct FINO 1. We were now able to compare the datasets. In Exercise 2 - Energy Meteorology we identified the logarithmic wind speed profile as a good method to predict the vertical wind speed profile. Therefore we fitted the calculated LiDAR data and the given FINO 1 data with the help of logarithmic wind speed profile. The procedure for FINO 1 is shown following code-snippet:

```
logProfileModel = @(b,z) b(1)/0.4 *(log(z/b(2)));
opts = statset('nlinfit');
opts.RobustWgtFun = 'bisquare';
logProfileCoeffsFino = real(nlinfit([33,40,50,60,70,80,90,100],...
avg_horSpeed_Fino,logProfileModel,[0.1,10^-5],opts));
```

The results are shown in figure 6 :

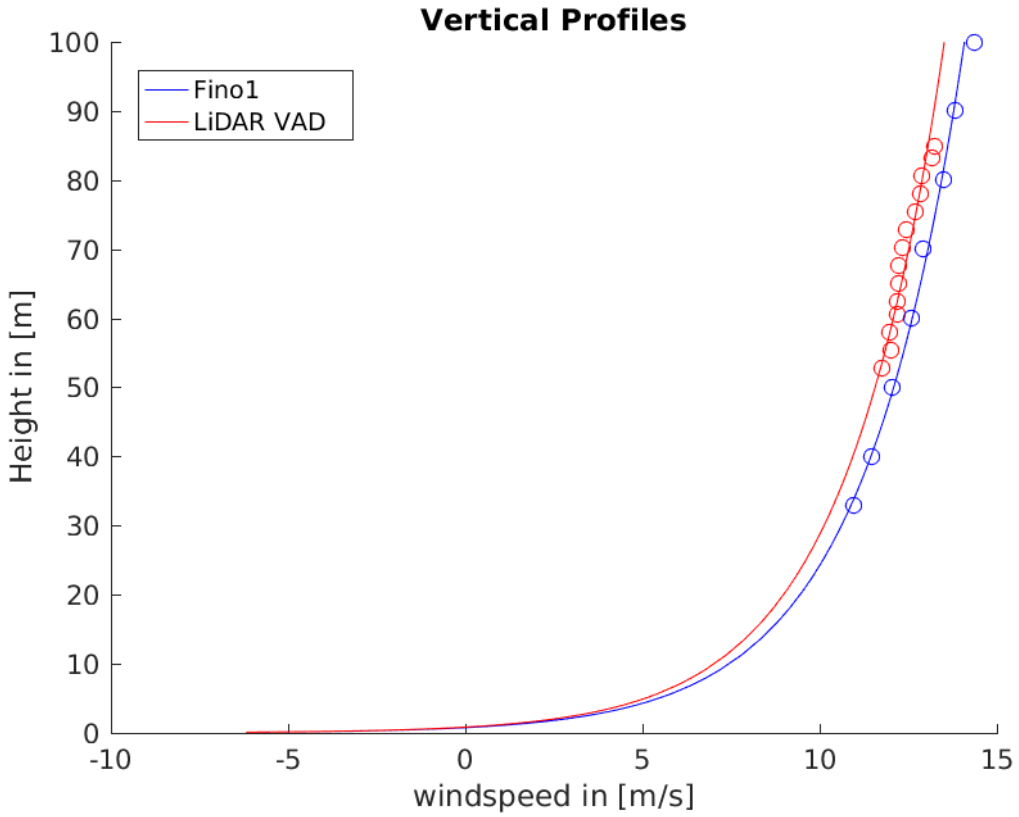


Figure 6: Vertical profile of FINO 1 and LiDAR measurement

The figure shows that the vertical profile of the LiDAR is shifted to the left. As discussed in the lecture, anemometers measure more accurately and therefore the FINO 1 curve should be more representative.

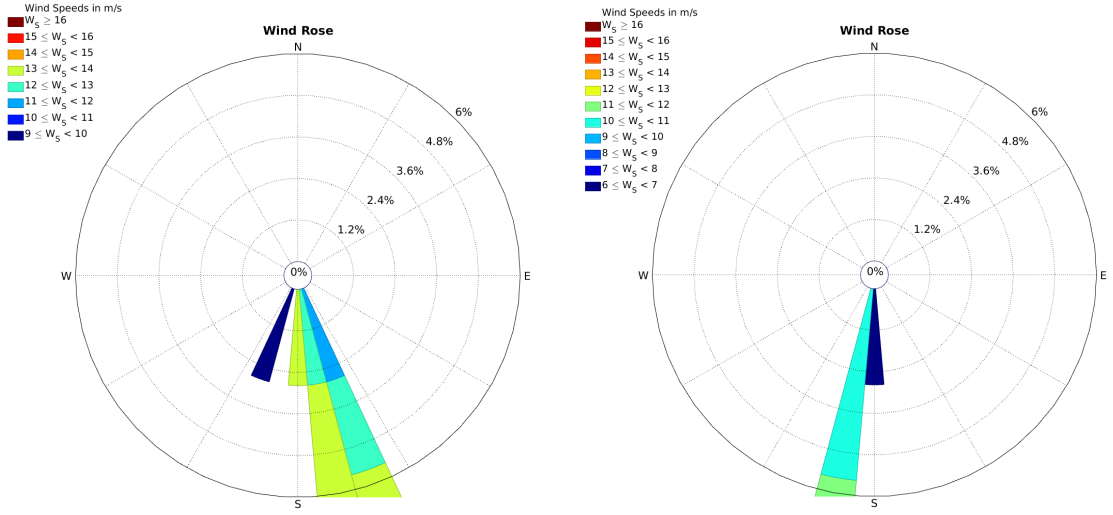
Further we evaluated the wind directions. For better comparison , we used measurements at similar heights. We used the already existing WindRose.m routine. The wind rose helps to evaluate

and compare the wind directions.

In order to obtain correct wind directions we used the following plot routine:

```
WindRose(fino_90_dir_interval,fino_speeds(:,7),'AngleNorth',0,'AngleEast',90,'freqlabelangle',45,'MaxFrequency',6);
WindRose(tenMinAvg_direction(:,7), tenMinAvg_horSpeed(:,7),'AngleNorth',0,'AngleEast',90,'freqlabelangle',45,'MaxFrequency',6);
```

The comparison between the two wind roses are shown in figure 7.



(a) Wind rose of FINO 1

(b) Wind rose of LiDAR VAD

Figure 7: Comparison of wind roses

We identify that there is noticeable shift in the wind directions from FINO 1 to the LiDAR VAD. If we only compare the 10-minute direction intervals the shift can be estimated to around 20°. One reason for the shift might be that we assume a homogeneous wind field. In reality we deal with fluctuations that influence the measurements.

3 Multi-Lidar 3D vector reconstruction: from three line-of-sight velocities to 3D wind vector

In this task we were provided a data set of about 90 minutes, measured by three short range WindScanners of the Technical University of Denmark. T

3.1 Data structure

he lidar data contains the 3D coordinates of each lidar system and the measured line-of-sight velocities. The three Lidars were focussed at the same measurement point at 90 meters height. The staring point was also give in 3D coordinates. The measurement setup is shown in figure 8.

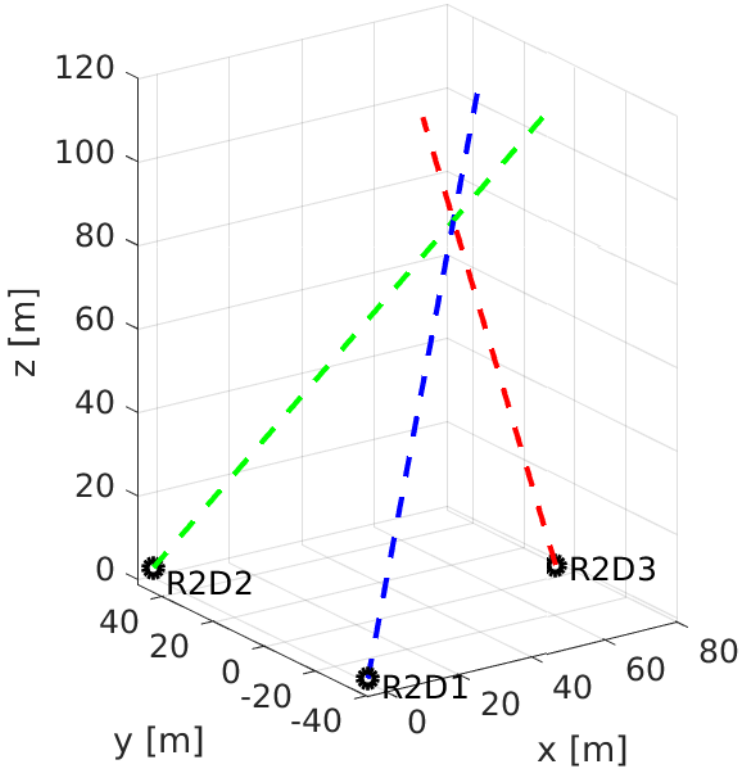


Figure 8: Measurement setup

3.2 Azimuth and elevation angle for each lidar

In order to calculate the the azimuth and the elevation we used the geometric set-up of the system. It should be noted that 0° azimuth angle is defined to be aligned with the positive x-axis and is counted counter-clockwise for positive values. The Matlab implementation is shown below:

```
for i=1:3
    distOnGround = sqrt((staring_point(1) - lidar_positions(1,i))^2+(staring_point
    (2) - lidar_positions(2,i))^2);
    distDiagonal = sqrt(distOnGround^2 + (staring_point(3) - lidar_positions(3,i))
    ^2);
    ele(i) = acosd(distOnGround/distDiagonal);
    azi(i) = atand((staring_point(2)-lidar_positions(2,i))/(staring_point(1)-
    lidar_positions(1,i)));
end

azi(3) = azi(3) + 180
```

The resulting angles are shown in the following table:

	elevation angle in °	azimuth angle in °
Lidar 1	56.6	35.93
Lidar 2	51.92	-46.04
Lidar 3	70.38	189.61

3.3 Matrix system that converts the three line-of-sight velocities to the u,v and w components

To calculate the three components of the line-of-sight velocities we had to set up a linear equation system in the form of $[A] \cdot b = c$. Each measured line of sight velocity is defined as:

$$V_{rad} = [u, v, w] \cdot [\cos(\beta)\sin(\alpha), \cos(\beta)\cos(\alpha), \sin(\beta)]$$

That leaves us with 3 equation for 3 unknown. The trigonometric functions represent the coefficient matrix A and the three different line of sight velocities V_{rad} represent c. To solve for $[u, v, w]$ we have to multiply A^{-1} with the radial wind speed vector. In Matlab this can be done with *mldivide*():

```
    solution = mldivide(matrix, V(i,:)');
    %solution = inv(matrix)*V(i,:)';
    u(i,1) = solution(1);
    v(i,1) = solution(2);
    w(i,1) = solution(3);
end
```

The calculated θ and δ are $0,089$ and $-0,022$.

3.4 Statistics

In the last section, the lidars are aligned with the main wind direction such that both v and w have a mean of 0 m/s. After that the mean and the standard deviation for each of the 3 lidars are calculated. The results are shown in the following table.

	μ	σ
Lidar 1	-4.83	0.76
Lidar 2	-3.20	0.63
Lidar 3	3.38	0.96

A Appendix

A.1 Spectral_Analysis.m

```
%% Spectra_Data
%  Script that converts raw spectra from the SpinnerLidar to the
%  line-of-sight wind speeds

close all; clc;

%% Reading the line-of-sight data of the SpinnerLidar

spinnerlidar_data    = dlmread('SpinnerLidar_Data_1s.txt');
spinnerlidar_spectra = dlmread('SpinnerLidar_Spectra_1s.txt');

index       = spinnerlidar_data(:,1);  % Lidar measurement index
vlos        = spinnerlidar_data(:,3);  % Line-of-sight measurement
sx          = spinnerlidar_data(:,7);  % Laser pointing unit vector x-component
sy          = spinnerlidar_data(:,8);  % Laser pointing unit vector y-component
focus       = spinnerlidar_data(:,9);  % Focus distance of Lidar

sz          = sqrt(1-sx.^2-sy.^2);
x           =  sz.*focus;
y           = -sy.*focus;
z           =  sx.*focus;

%% Reading the spectrum data of the SpinnerLidar

bins     = 256;             % amount of bins in each spectrum
bandwith = 25e6;            % frequency bandwith of the spectra
lambda   = 1560e-9;         % wavelength of the laser light

%% Step 1: noise cancelling by discarding all bins below mean*1.1 per spectra
for pos = 1:312

    spinner_noiseCancelled(pos,:) = spinnerlidar_spectra(pos,:)-1.5*mean(
    spinnerlidar_spectra(pos,:));
    spinner_noiseCancelled(spinner_noiseCancelled < 0) = 0;
%    figures created once in directory
%    figure('visible','off')
%    plot(spinner_noiseCancelled(pos,:));
%    saveas(gcf,strcat('figures/spectra_noisecancels_normed_',num2str(pos) ,'.jpg'))
    ;
end;

%% Step 2
```

```
% Calculate the frequency resolution of the spectra, the Doppler frequency of each
% bin and from that the wind speed corresponding to each bin:
c = 3*10^8;
f_0 = c/lambda;
lambda_0 = c/f_0;
for bin=1:256
    f_d(bin,1) = (bin-1)/bins*bandwith;
    v(bin,1) = f_d(bin,1) *lambda_0; %divided by 2?
end;

%% Finding the spectra peaks and visualize
%%Step 3: apply centroid method
for pos = 1:312
    spinnerSum = sum(spinner_noiseCancelled(pos,:))
    spinner_noiseCancelled_normed(pos,:) = spinner_noiseCancelled(pos,:)/spinnerSum;
end;

for pos = 1:312
    f_peak(pos,1) = sum(spinner_noiseCancelled_normed(pos,:)*f_d(:))/sum(
    spinner_noiseCancelled_normed(pos,:));
end;

%detect centroid failures
for pos = 1:312
    maxIndex= find(spinner_noiseCancelled_normed(pos,:) == max(
    spinner_noiseCancelled_normed(pos,:)));
    centroidIndex= f_peak(pos)/bandwith*bins;
    failures(pos,1) = abs(centroidIndex-maxIndex);
end;
figure();
hold on;
x_range = 1:length(failures)
% failures_mean = nanmean(failures);
% failures_corrected = failures - failures_mean;
% failures(failures_corrected < 0) = 0;
plot(x_range,failures);
for k=1:length(failures)
      if failures(k) > 5
      text(x_range(k),failures(k),num2str(x_range(k)))
      end
end
%[pks,locs] = findpeaks(failures);
%text(locs+.02,pks,num2str((1:numel(pks))'));
xlabel('Spectra number','fontsize',10)
ylabel('Differences of bin indices for max and centroid functions','fontsize',10)
saveas(gcf,strcat('figures/failures.png'));
hold off;
```

```
86
%%Step 4: Calc centroid velocity for correlation with lidar measured speeds
88 vlos_centroid(:,1) = f_peak(:,1) *lambda_0; %divided by 2?

90 %% Plotting
figure();
92 plot(spinnerlidar_spectra(1,:))
xlabel('Frequency')
94 ylabel('Collection efficiency')
title('Spectrum without filter')
96 saveas(gcf,strcat('figures/spectra_nofilter.png'));

98 figure();
plot(spinner_noiseCancelled(1,:))
100 title('Spectrum with filter')
xlabel('Frequency')
102 ylabel('Collection efficiency')
saveas(gcf,strcat('figures/spectra_noisecanceld.png'));
104
check = vlos-vlos_centroid;
106 Correlation(vlos,vlos_centroid);

108 Rosette_Scan_Plot(y,z,vlos_centroid,...
        'coloraxis',[5 10],...
110         'rosettebackground',y,z,...
        'meshgridvector',(-1:0.1:1)*unique(focus)/2);
112
Rosette_Scan_Plot(y,z,vlos,...
114         'coloraxis',[5 10],...
        'rosettebackground',y,z,...
116         'meshgridvector',(-1:0.1:1)*unique(focus)/2);
```

A.2 Vad_analysis.m

```
%% Loading

close all; clc;
load('201401020600_WLS200S-17_data_.mat');

ts_VAD = data.ts;
az_c_VAD = data.az_c;
az_c_VAD = az_c_VAD + 180;
rs_VAD = data.rs;
el_VAD = data.el;
rg_VAD = data.rg;
x_VAD = data.x;
y_VAD = data.y;
z_VAD = data.z;
cnr_VAD = data.cnr;

%% Step 1
% Write Data to one double struct
data_VAD = [ts_VAD az_c_VAD rs_VAD cnr_VAD el_VAD rg_VAD x_VAD y_VAD z_VAD];

% Filter data by CnR, by setting to NaN and later in Step 2 discarding NaN
% values
cnr_ind = find(cnr_VAD <= -20 | cnr_VAD >= -5);
rs_VAD(cnr_ind) = NaN;
az_c_VAD(cnr_ind) = NaN;


% get all range gates
count = 1;

for q = 1:100
    a = find(rg_VAD == q);
    if length(a) ~= 0
        rg_ind(:,count) = a;
        rangeGateSet(count)=q;
        count = count + 1;
    end
end

% separate data by range gates
for i = 1:14
    rs_VAD_filter(:,i) = data_VAD(rg_ind(:,i),3);
    az_c_VAD_filter(:,i) = data_VAD(rg_ind(:,i),2);
    ts_VAD_filter(:,i) = data_VAD(rg_ind(:,i),1);
end
```

```
46
% how long does one full 360 scan take?
48 time_per_scan = 360 / 25;
intervalls_per_scan = time_per_scan / 0.4;
50
%% Step 2
52 VADCos = @(param,phi) param(1)*cosd(phi-param(2))+param(3);
startvalues = [2, 10, 2];
54 numberOfScans = floor(length(az_c_VAD_filter)/intervalls_per_scan-1)
calc_times = zeros(numberOfScans,14);
56 calc_vHor = zeros(numberOfScans,14);
calc_vVer = zeros(numberOfScans,14);
58 calc_D = zeros(numberOfScans,14);

60 for j = 1:14 % number of Range Gates
    for i = 1:numberOfScans
62      phi = az_c_VAD_filter((i-1)*intervalls_per_scan+18:(i-1)*intervalls_per_scan
    +53,j);
        v_r = rs_VAD_filter((i-1)*intervalls_per_scan+18:(i-1)*intervalls_per_scan
    +53,j);
64      nanIndices = isnan(phi) | isnan(v_r);
        phi(nanIndices) = [];
66      v_r(nanIndices) = [];
        if length(phi>0)
68          fitparam = lsqcurvefit(VADCos, startvalues, phi, v_r);
            savedparam{i,j} = fitparam;
70          %calculate horiz and vertical speed
            v_hor = fitparam(1) / cosd(60);
72          v_ver = -fitparam(3) / sind(60);
            D=(fitparam(1)+180);
74          calc_times(i,j) = ts_VAD_filter((i-1)*intervalls_per_scan+18,j);
            calc_vHor(i,j) =  v_hor;
76          calc_vVer(i,j) = v_ver;
            calc_D(i,j) = D;
78      end
    end
80 end


82
%% Step 3: compare lidar and sonic data
84 startTime = datenum('02-Jan-2014 06:00:00','dd-mmm-yyyy HH:MM:SS');
endTime = datenum('02-Jan-2014 12:00:00','dd-mmm-yyyy HH:MM:SS');
86
%10 minute averages of the lidar wind speed (horizontal)
88 for j=1:14
    for interval = 1:36 %we have 36 ten minute intervals
```

```
90          tenMinAvg_horSpeed(interval,j) = nanmean(calc_vHor(calc_times(:,j) >=
        startTime +(interval-1)/(24*6) & ...
                                                    calc_times(:,j) < startTime +
        interval/(24*6),j));
92          tenMinAvg_direction(interval,j) = nanmean(calc_D(calc_times(:,j) >=
        startTime +(interval-1)/(24*6) & ...
                                                    calc_times(:,j) < startTime +
        interval/(24*6),j));
94      end;
    end;
96
    %load fino1 data
98  fino_33 = readtable('fino-150504160459/
        FINO1_Windgeschwindigkeit_33m_20131220_20140121.dat','Delimiter','tab','
        HeaderLines',6);
    fino_40 = readtable('fino-150504160459/
        FINO1_Windgeschwindigkeit_40m_20131220_20140121.dat','Delimiter','tab','
        HeaderLines',6);
100 fino_50 = readtable('fino-150504160459/
        FINO1_Windgeschwindigkeit_50m_20131220_20140121.dat','Delimiter','tab','
        HeaderLines',6);
    fino_60 = readtable('fino-150504160459/
        FINO1_Windgeschwindigkeit_60m_20131220_20140121.dat','Delimiter','tab','
        HeaderLines',6);
102 fino_70 = readtable('fino-150504160459/
        FINO1_Windgeschwindigkeit_70m_20131220_20140121.dat','Delimiter','tab','
        HeaderLines',6);
    fino_80 = readtable('fino-150504160459/
        FINO1_Windgeschwindigkeit_80m_20131220_20140121.dat','Delimiter','tab','
        HeaderLines',6);
104 fino_90 = readtable('fino-150504160459/
        FINO1_Windgeschwindigkeit_90m_20131220_20140121.dat','Delimiter','tab','
        HeaderLines',6);
    fino_100 = readtable('fino-150504160459/
        FINO1_Windgeschwindigkeit_100m_20131220_20140121.dat','Delimiter','tab','
        HeaderLines',8);
106 fino_speeds(:,1) = fino_33.Var2((datenum(fino_33.Var1(:)) >= startTime & datenum(
        fino_33.Var1(:)) < endTime));
    fino_speeds(:,2) = fino_40.Var2((datenum(fino_40.Var1(:)) >= startTime & datenum(
        fino_40.Var1(:)) < endTime));
108 fino_speeds(:,3) = fino_50.Var2((datenum(fino_50.Var1(:)) >= startTime & datenum(
        fino_50.Var1(:)) < endTime));
    fino_speeds(:,4) = fino_60.Var2((datenum(fino_60.Var1(:)) >= startTime & datenum(
        fino_60.Var1(:)) < endTime));
110 fino_speeds(:,5) = fino_70.Var2((datenum(fino_70.Var1(:)) >= startTime & datenum(
        fino_70.Var1(:)) < endTime));
```

```
fino_speeds(:,6) = fino_80.Var2((datenum(fino_80.Var1(:)) >= startTime & datenum(
    fino_80.Var1(:)) < endTime));
fino_speeds(:,7) = fino_90.Var2((datenum(fino_90.Var1(:)) >= startTime & datenum(
    fino_90.Var1(:)) < endTime));
fino_speeds(:,8) = fino_100.Var2((datenum(fino_100.Var1(:)) >= startTime & datenum(
    fino_100.Var1(:)) < endTime));
fino_speeds(fino_speeds==-999) = NaN;
%overall mean speeds for lidar
for j=1:14
    avg_horSpeed_Lidar(j) = nanmean(tenMinAvg_horSpeed(:,j));
end;
%overall mean speeds for fino
avg_horSpeed_Fino(1) = nanmean(fino_speeds(:,1));
avg_horSpeed_Fino(2) = nanmean(fino_speeds(:,2));
avg_horSpeed_Fino(3) = nanmean(fino_speeds(:,3));
avg_horSpeed_Fino(4) = nanmean(fino_speeds(:,4));
avg_horSpeed_Fino(5) = nanmean(fino_speeds(:,5));
avg_horSpeed_Fino(6) = nanmean(fino_speeds(:,6));
avg_horSpeed_Fino(7) = nanmean(fino_speeds(:,7));
avg_horSpeed_Fino(8) = nanmean(fino_speeds(:,8));


%perform power log fit for fino1
logProfileModel = @(b,z) b(1)/0.4 *(log(z/b(2)));
opts = statset('nlinfit');
opts.RobustWgtFun = 'bisquare';
logProfileCoeffsFino = real(nlinfit([33,40,50,60,70,80,90,100],avg_horSpeed_Fino,
    logProfileModel,[0.1,10^-5],opts));
lidarHeights = rangeGateSet*sind(60);
logProfileCoeffsLidar= real(nlinfit(lidarHeights,avg_horSpeed_Lidar,logProfileModel
    ,[0.1,10^-5],opts));
%%
%plot both vertical wind speed profiles
x_range = [33,40,50,60,70,80,90,100]
figure();
hold on;
[xLogFino,yLogFino]=fplot(@(z) logProfileCoeffsFino(1)/0.4 *(log(z/
    logProfileCoeffsFino(2))),[0 100]);
[xLogLidar,yLogLidar]=fplot(@(z) logProfileCoeffsLidar(1)/0.4 *(log(z/
    logProfileCoeffsLidar(2))),[0 100]);
plot(yLogFino,xLogFino,'Color','b');
plot(yLogLidar,xLogLidar,'Color','r');
plot(avg_horSpeed_Lidar,lidarHeights,'or');
plot(avg_horSpeed_Fino,x_range, 'ob');
ylabel('Height in [m]');
xlabel('windspeed in [m/s]');
title('Vertical Profiles');
```

```
legend('Fino1','LiDAR VAD','Location','northwest');
saveas(gcf,'figures/verticalProfiles.png')
hold off;

%% Winddirections
fino_90_dir = readtable('fino-150504160459/FINO1_Windrichtung_90m_20131220_20140121.dat','Delimiter','tab','HeaderLines',6);
fino_90_dir_interval = fino_90_dir.Var2((datenum(fino_90_dir.Var1(:)) >= startTime & datenum(fino_90_dir.Var1(:)) < endTime));

WindRose(fino_90_dir_interval,fino_speeds(:,7),'AngleNorth',0,'AngleEast',90,'freqlabelangle',45,'MaxFrequency',6);
saveas(gcf,'figures/WindRose_Fino1.png')
WindRose(tenMinAvg_direction(:,7), tenMinAvg_horSpeed(:,7),'AngleNorth',0,'AngleEast',90,'freqlabelangle',45,'MaxFrequency',6);
saveas(gcf,'figures/WindRose_lidar.png')
figure();
hold on;
plot(fino_90_dir_interval);
plot(tenMinAvg_direction);
hold off;


%% Plots for report
figure();
hold on;
plot(VADCos(savedparam{1,1},1:360));
scatter(az_c_VAD_filter(18:53,1), rs_VAD_filter(18:53,1));
hold off;
xlabel('azimuth angle [degree]')
ylabel('V_{los} [m/s]')
saveas(gcf,'figures/fit.png')
```

A.3 Multi_lidar_3D_reconstruction

```
close all; clc;

% Measurement geometry [x, y, z] of the measurement point and the three
% Lidars

staring_point    =   [ 46.8,     0, 90]';
lidar_positions  =   [-0.94,-34.59, 0.60;
                      -1.01, 49.58, 2.09;
                      78.69,  5.40, -0.74]';

lidar_range=sqrt((repmat(staring_point(1),1,3)-lidar_positions(1,:)).^2+...
          (repmat(staring_point(2),1,3)-lidar_positions(2,:)).^2);

% Here you have to calculate the azimuth and elevation scanning angles of each of
% the lidars:
for i=1:3
    distOnGround = sqrt((staring_point(1) - lidar_positions(1,i))^2+(staring_point
    (2) - lidar_positions(2,i))^2);
    distDiagonal = sqrt(distOnGround^2 + (staring_point(3) - lidar_positions(3,i))
    ^2);
    ele(i) = acosd(distOnGround/distDiagonal);
    azi(i) = atand((staring_point(2)-lidar_positions(2,i))/(staring_point(1)-
    lidar_positions(1,i)));
end
azi(3) = azi(3) + 180
% Reading the line-of-sight velocity data

vlos1 = csvread('R2D1.csv',1,0);
vlos2 = csvread('R2D2.csv',1,0);
vlos3 = csvread('R2D3.csv',1,0);

vlos1(:,1)=vlos1(:,1)/1000;     % First column is time in milliseconds
vlos2(:,1)=vlos2(:,1)/1000;     % First column is time in milliseconds
vlos3(:,1)=vlos3(:,1)/1000;     % First column is time in milliseconds

fs = 97.65625;  % Sampling frequency
dt = 1/fs;      % Time step
starttime = dt;
endtime   = min([vlos1(end,1) vlos2(end,1) vlos3(end,1)]);
timeline = (starttime:dt:endtime)';

% The line-of-sights speeds are interpolated to the same timeline to be
% perfectly synchronised. Continue working with these vectors:

vlos1_interp = interp1(vlos1(:,1),vlos1(:,2),timeline);
```

```
vlos2_interp = interp1(vlos2(:,1),vlos2(:,2),timeline);
vlos3_interp = interp1(vlos3(:,1),vlos3(:,2),timeline);

% Conversion of the three line-of-sight speeds to u and v

% matrix = ...
% u = ...
% v = ...
% w = ...

vlos1_interp = (vlos1_interp);
vlos2_interp = (vlos2_interp);
vlos3_interp = (vlos3_interp);



matrix = [cosd(ele(1))*cosd(azi(1)),cosd(ele(1))*sind(azi(1)),sind(ele(1));...
            ,cosd(ele(2))*cosd(azi(2)),cosd(ele(2))*sind(azi(2)),sind(ele(2));...
            ,cosd(ele(3))*cosd(azi(3)),cosd(ele(3))*sind(azi(3)),sind(ele(3))];

V = [vlos1_interp vlos2_interp vlos3_interp];
for i = 1:length(vlos1_interp)
    solution = mldivide(matrix, V(i,:)');
    %solution = inv(matrix)*V(i,:)';
    u(i,1) = -solution(1);
    v(i,1) = -solution(2);
    w(i,1) = -solution(3);
end
% In this part the calculated u, v and w components are aligned with the
% 'main' wind direction, such that both v and w have a mean of 0 m/s:

theta = atan(mean(v)/mean(u));
delta = -atan(mean(w)/sqrt(mean(u)^2+mean(v)^2));
alignmatrix1=Rotate_Matrix(delta*180/pi,'y');
alignmatrix2=Rotate_Matrix(theta*180/pi,'z');
V_aligned=alignmatrix2*alignmatrix1*V'; V_aligned=V_aligned';
statistics = [ mean(V_aligned);
               std(V_aligned)];

% Plotting of results

colors='bgr';
figure
hold on
 for i=1:3
    scatter3(lidar_positions(1,i),...
          lidar_positions(2,i),...
```

```
            lidar_positions(3,i),'k','linewidth',3)
        textstring=sprintf('%s%d','R2D',i);
        text(lidar_positions(1,i),...
            lidar_positions(2,i)-5,textstring,'fontsize',12);
            formatstring=sprintf('%s%s',colors(i),'--');
    plot3([lidar_positions(1,i),staring_point(1)*1.3-0.3*lidar_positions(1,i)],...
            [lidar_positions(2,i),staring_point(2)*1.3-0.3*lidar_positions(2,i)],...
            [lidar_positions(3,i),staring_point(3)*1.3-0.3*lidar_positions(3,i)],...
            formatstring,'linewidth',2);
end
grid on
axis equal
xlabel('x [m]','fontsize',14)
ylabel('y [m]','fontsize',14)
zlabel('z [m]','fontsize',14)
set(gca,'fontsize',12)
xlim([-5 80]); ylim([-40 50]); zlim([-1 120]);
view([-35 20])
saveas(gcf,'lidar.png')
```