

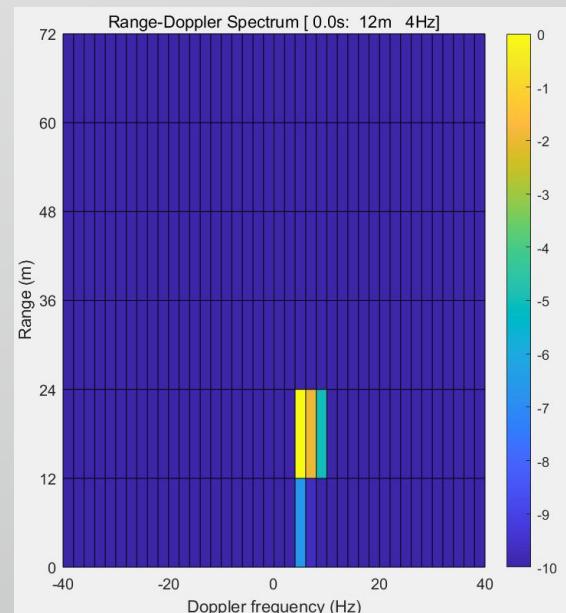
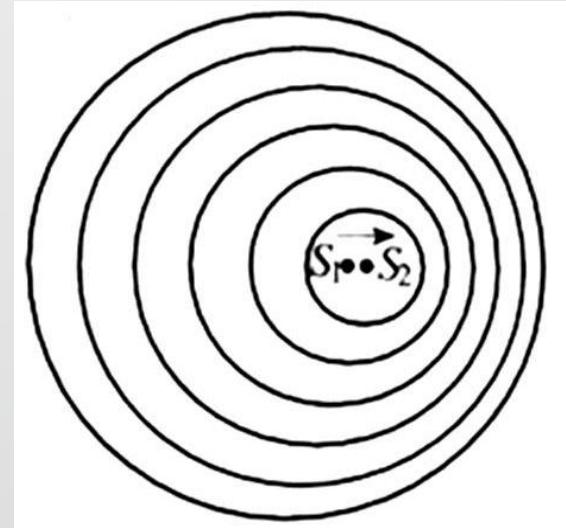
Project2 : Motion detection via communication signals



主讲老师：王小静
办公地点：一教131

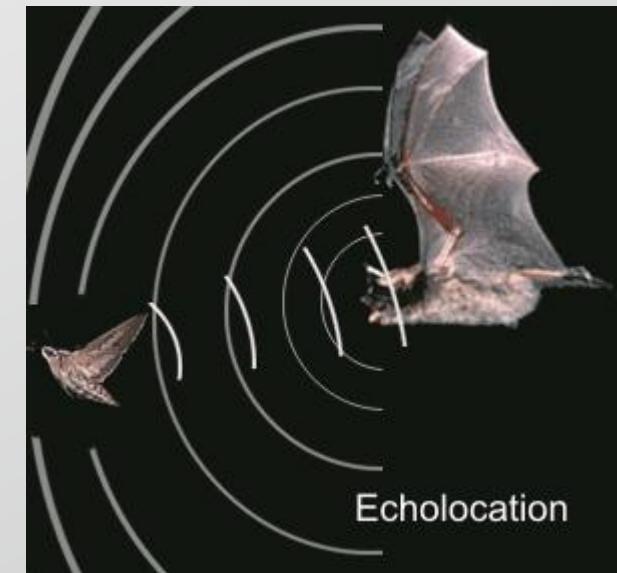
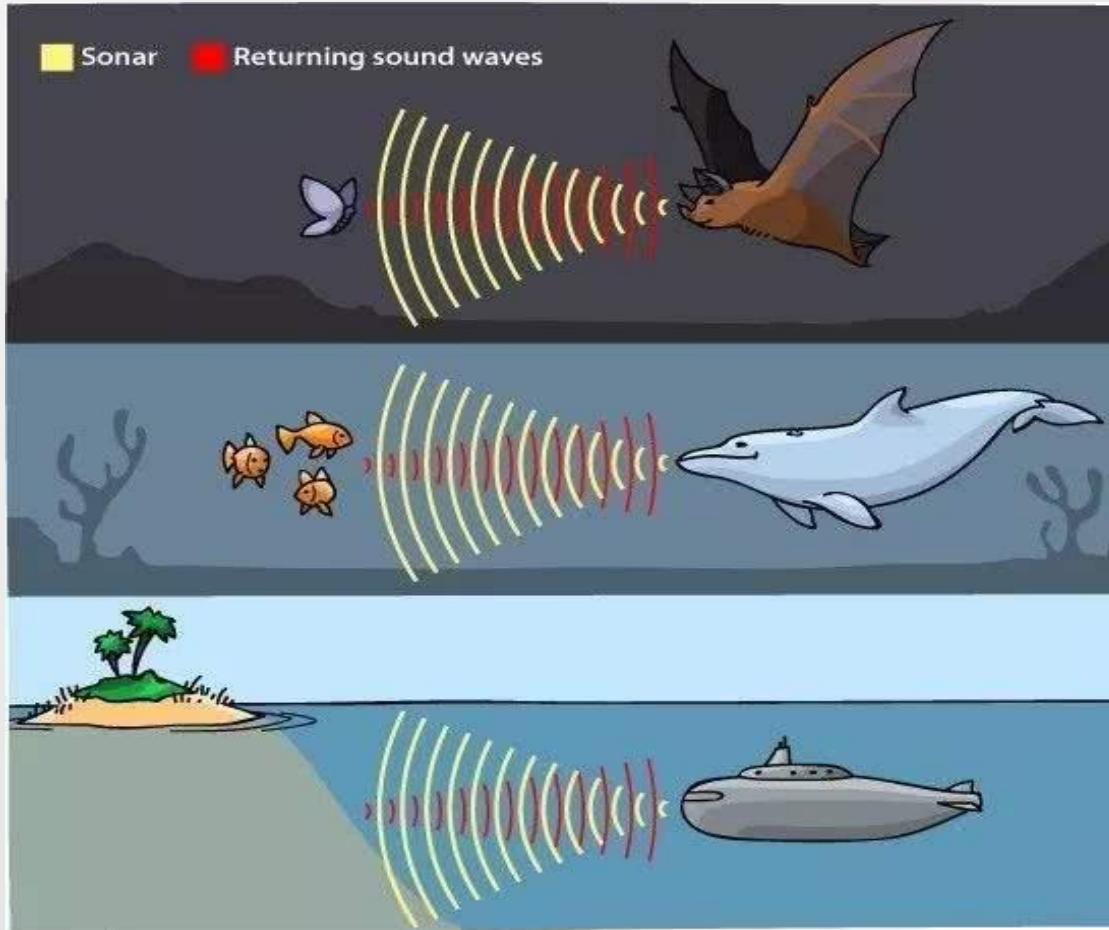
Overview

- **Part I: Background**
 - Target Localization by Rador
 - Multipath propagation
 - Doppler shift
- **Part II: Passive Rador Introduction**
 - The basic principle of Passive Rador
 - Cross-Correlation(ambiguity function)
 - Target Range/Doppler Tracking
- **Part III: Project Tasks**
 - Measurement Environment

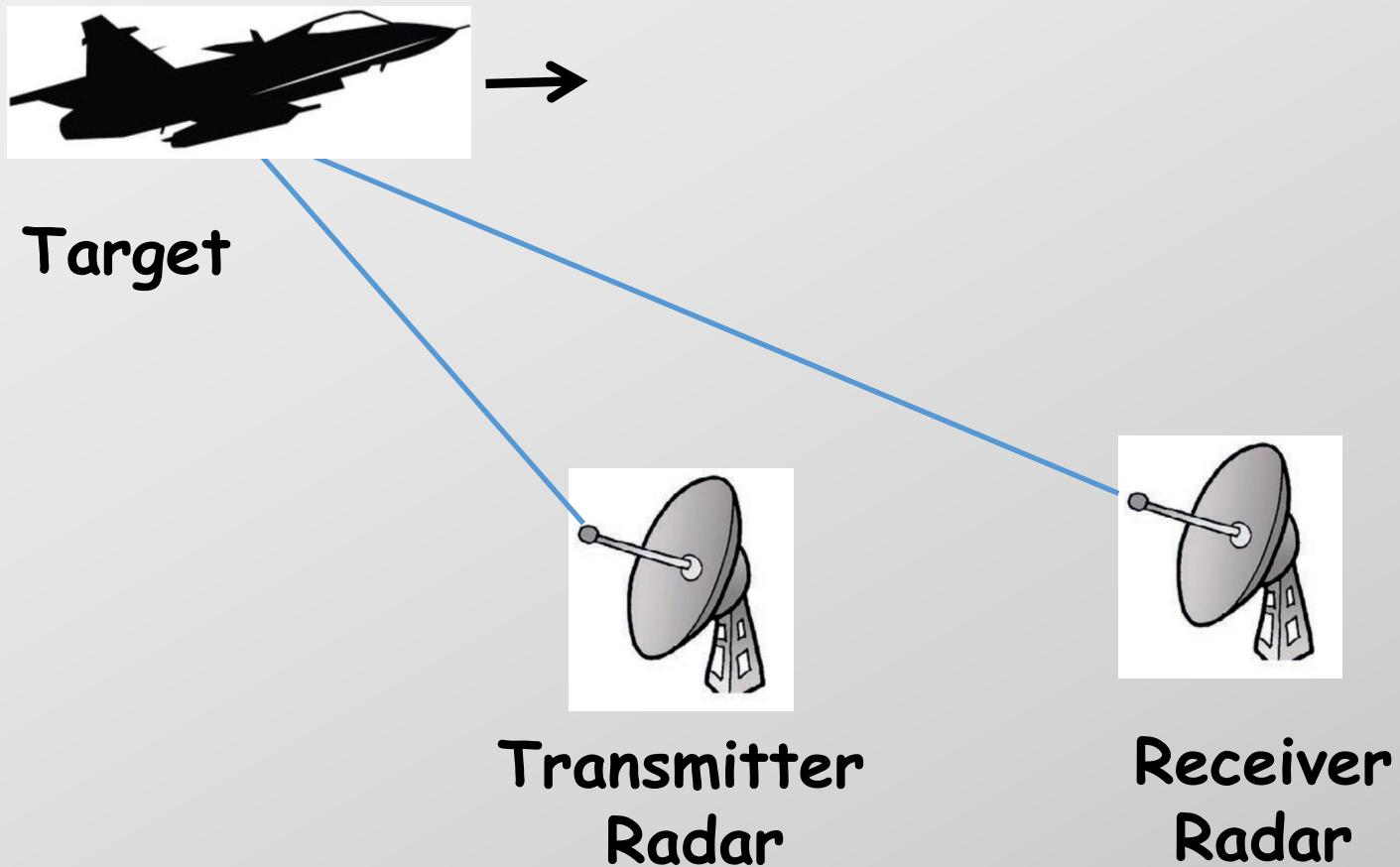


Part I: Backgound

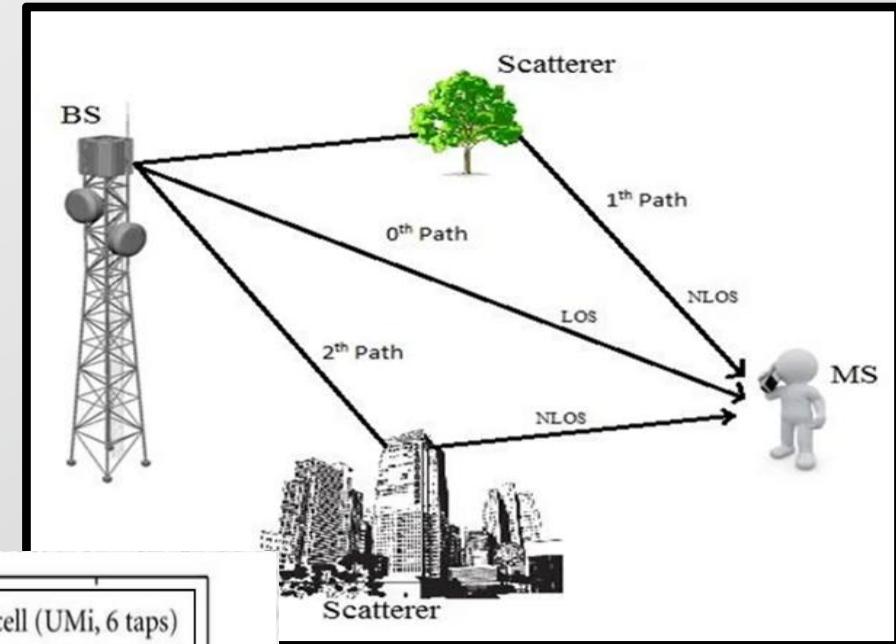
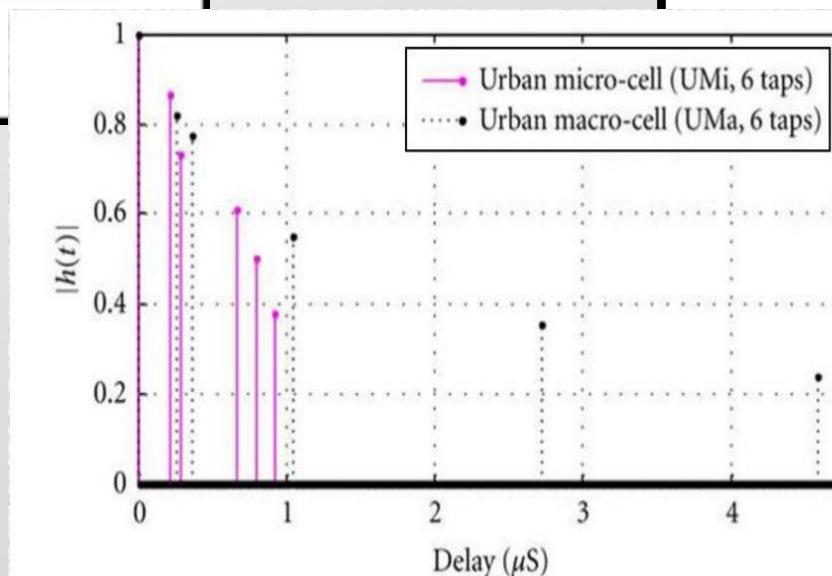
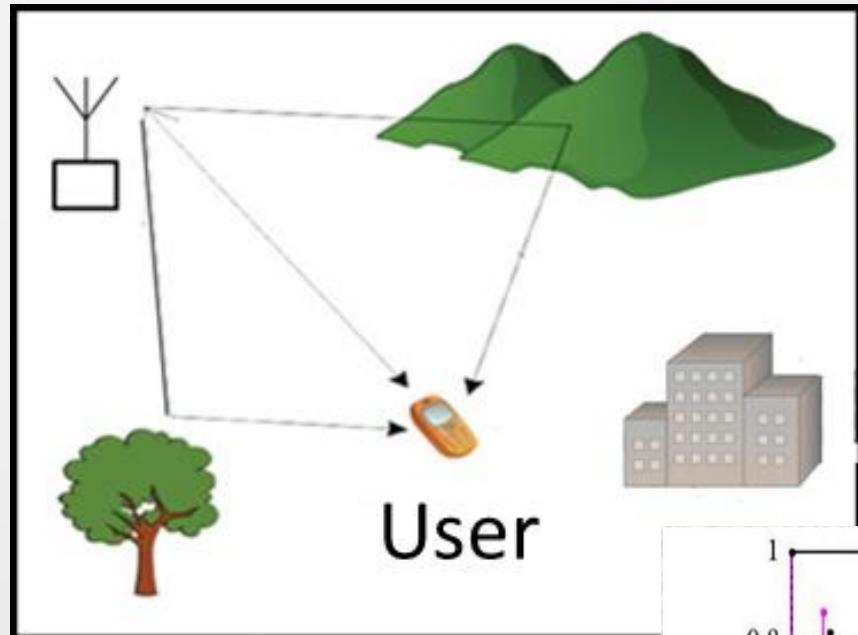
Target Localization



Active Radar

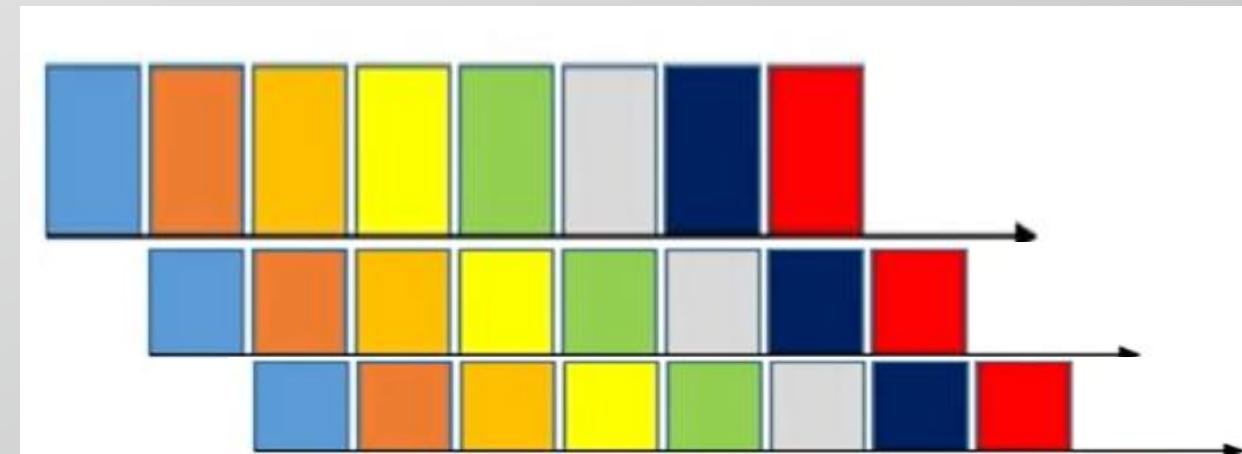
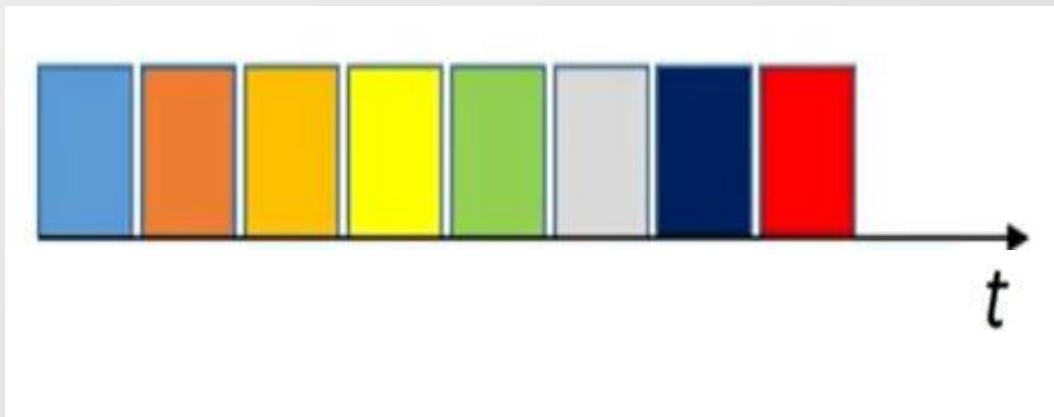
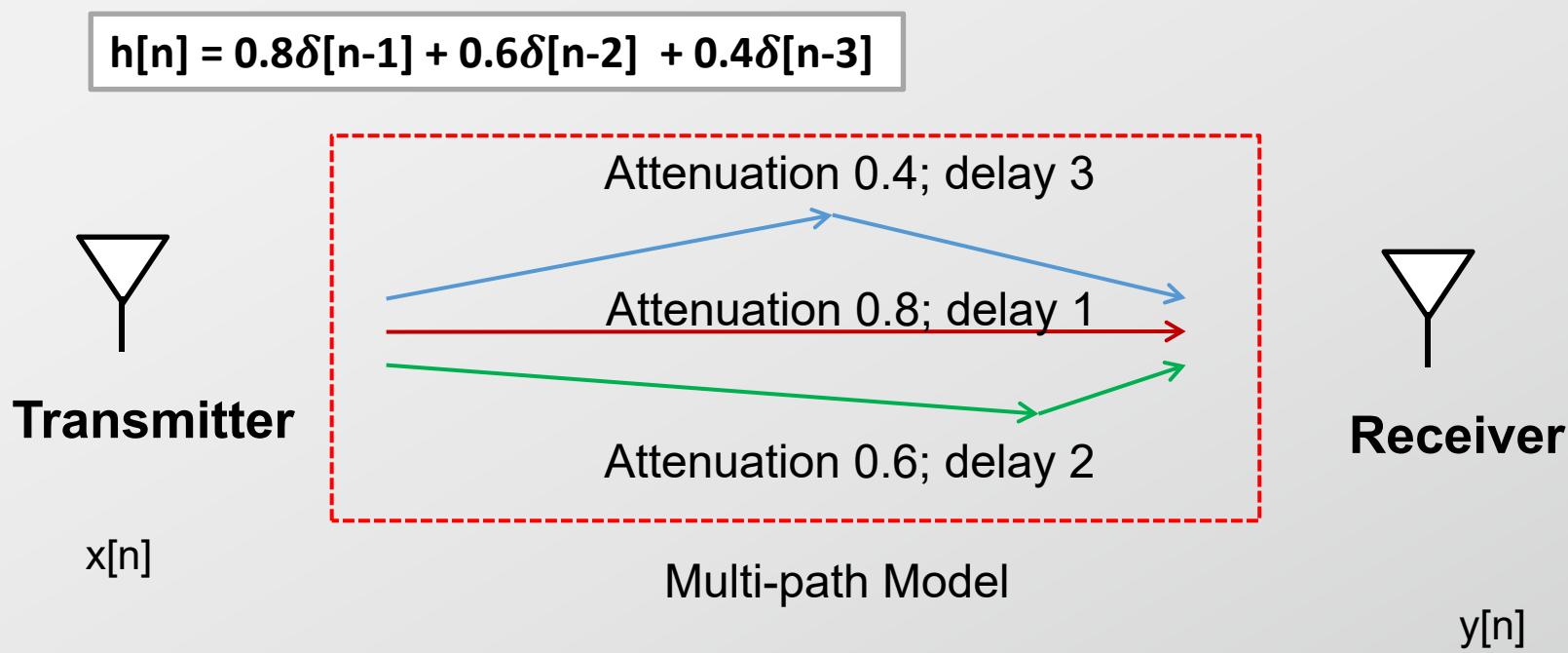


Multi-path propagation



Multi-path Delay Spread

Multi-path Model:



Doppler Shift



救护车的声音就是最好的例子

Am Krankenwagen kann man es eigentlich am besten hören.

The difference between the transmitted and received frequencies caused by object movement is called Doppler shift.

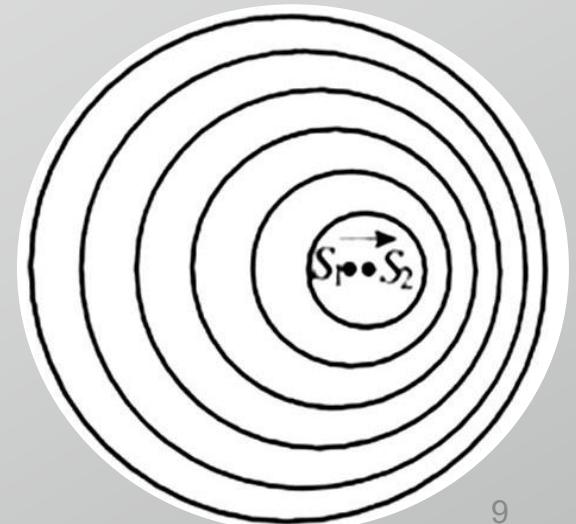
$$\Delta f = f' - f$$

f' : the frequency that's received

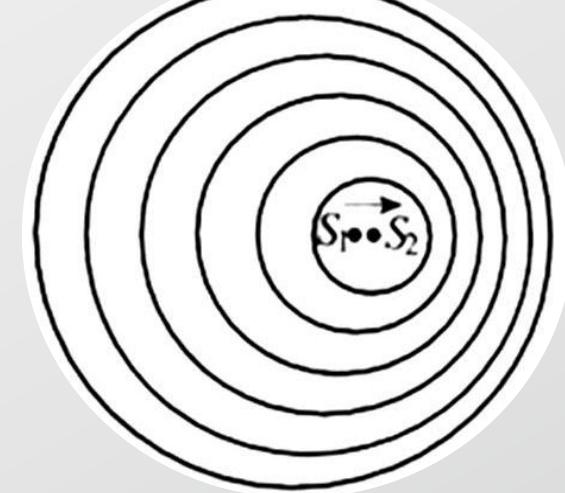
f : the frequency that's transmitted

Closer: $\Delta f > 0$;

Farther: $\Delta f < 0$.



$$f' = \left(\frac{v \pm v_o}{v \mp v_s} \right) f$$



f' : the frequency that's received;

f : the original frequency that's transmitted;

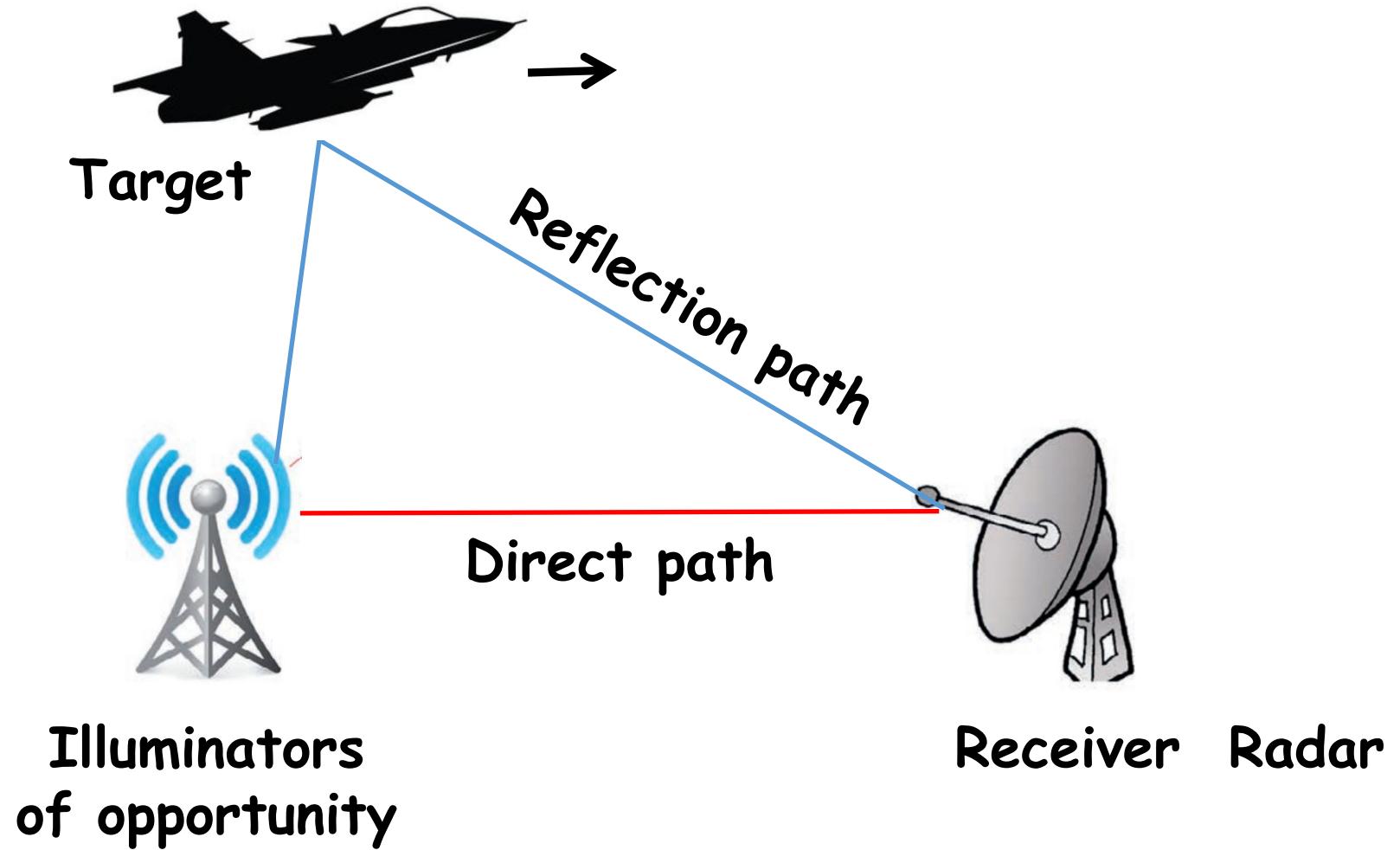
v : the travel speed of waves in the medium;

v_o : the moving speed of the receiver relative to the medium. If it is closer to the transmitter, the front operation symbol is + sign, otherwise it is - sign;.

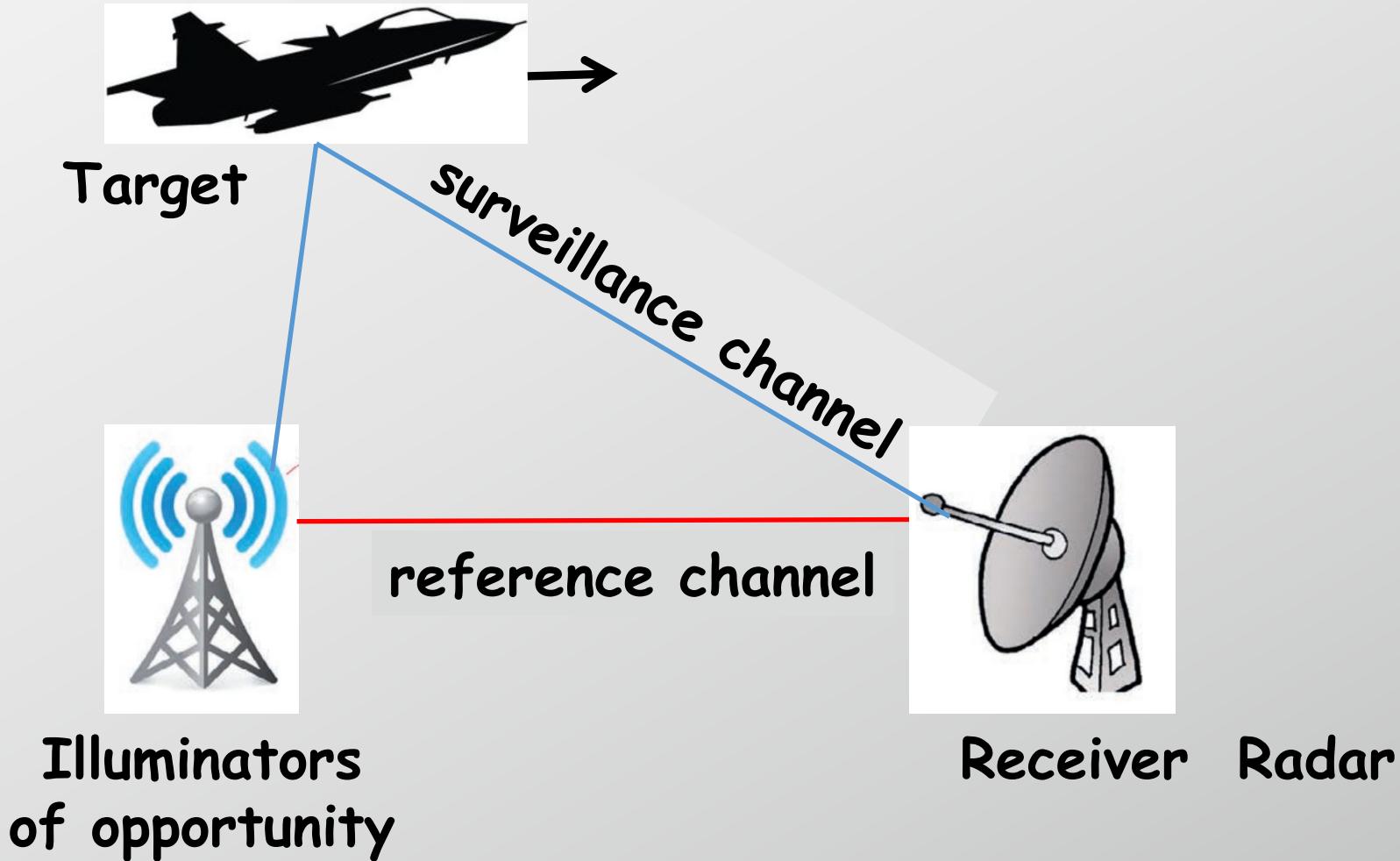
v_s : the moving speed of the transmitter relative to the medium. If it is closer to the receiver, the front operation symbol is - sign, otherwise it is + sign.

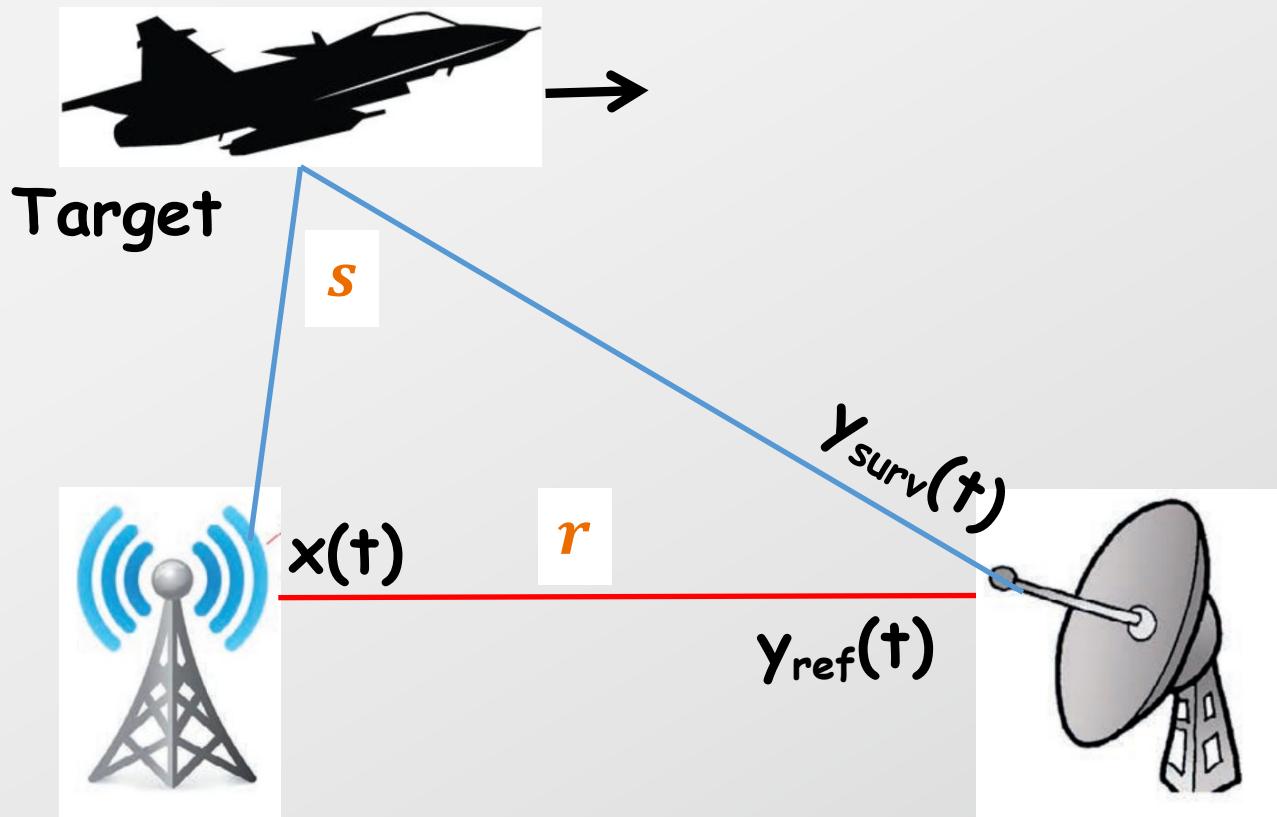
Part II: Passive Rador Introduction

Passive Radar



The basic principle of Passive Rador





Suppose the transmitted signal is $x(t)$

- The reference signal is the transmitted signal after time delay and attenuation:

$$y_{ref}(t) = \alpha x(t - \tau_r)$$

- The surveillance signal is the transmitted signal after attenuation, time delay and Doppler frequency offset:

$$y_{surv}(t) = \beta x(t - \tau_s) e^{j2\pi f t}$$

$$\tau_r = \frac{r}{c} \quad \tau_s = \frac{s}{c} \quad f = \frac{vf_c}{c}, \text{ where } v = \frac{ds}{dt}$$

Cross-Correlation(ambiguity function)

$$Cor(\tau, f_D) = \int_t^{t+T} y_{surv}(t)y_{ref}^*(t - \tau) e^{-j2\pi f_D t} dt$$

where $y_{surv}(t)$ is the complex envelope of the signal received in the surveillance channel, $y_{ref}(t)$ is a (cleaned) replica of the transmitted signal, τ is the potential TDOA of the target echo signal, and f_D is the potential corresponding bistatic Doppler. (TDOA:the Time Difference of Arrival between the reference signal and the surveillance echo.)

The cross-correlation between the reference signal and the surveillance signal, also referred to as Range-Doppler correlation.

Target Range/Doppler Tracking

$$Cor(\tau, f_D) = \int_t^{t+T} y_{surv}(t)y_{ref}^*(t - \tau) e^{-j2\pi f_D t} dt$$

- estimate (τ, f_D) :

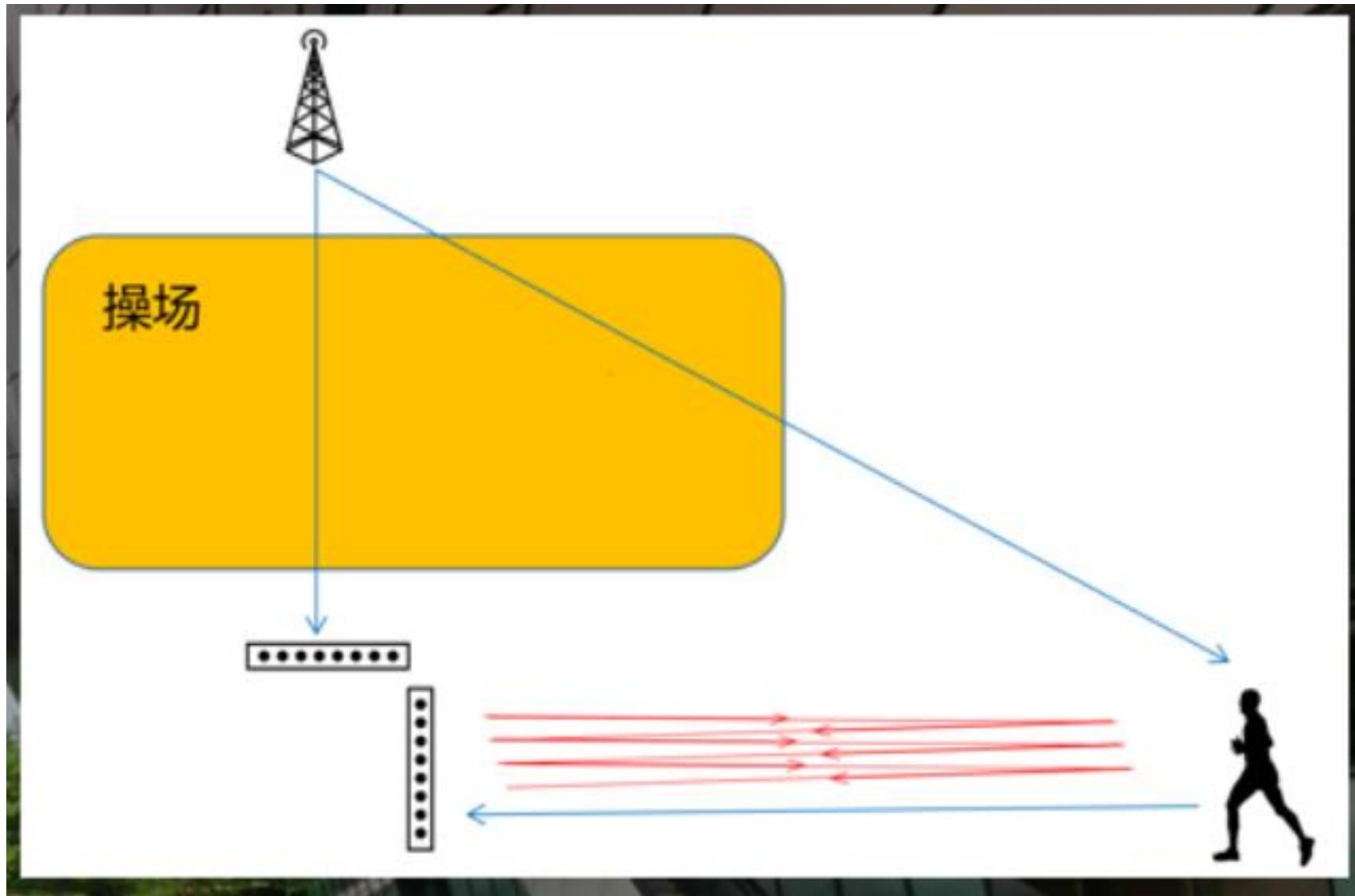
$$(\hat{\tau}, \widehat{f_D}) = \arg \max_{\tau, f_D} Cor(\tau, f_D)$$

- $\hat{\tau}$ is the estimate of $\tau_s - \tau_r$, $\widehat{f_D}$ is the estimate of f .

Part III: Project Tasks

Measurement Environment





Project Introduction

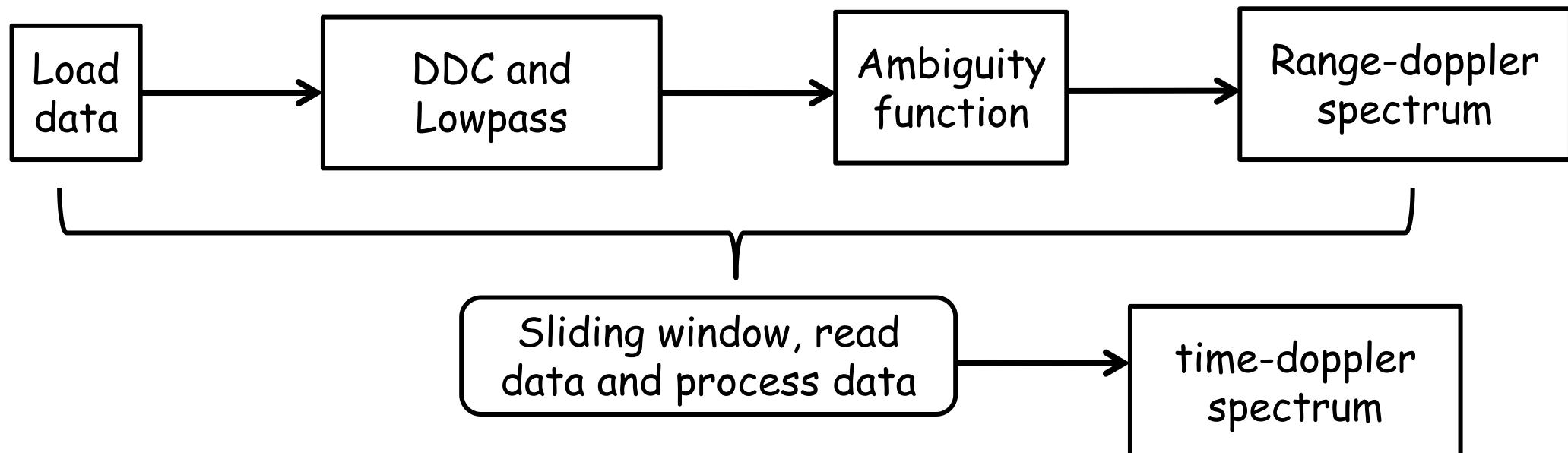
◆ **Data:** two signals, one is the reference signal, the other is the surveillance signal reflected from the moving human body.

◆ **Objective:**

- ✓ Basic: Estimate the $(\hat{\tau}, \hat{f}_D)$ for four sets of data and draw range-doppler spectrum.
- ✓ Optional: Obtain the relationship between the Doppler frequency caused by human motion and time, so as to figure out the mode of motion.

◆ **Instructions of Transmitter:** There are two base stations with bandwidth of 20MHz(2110-2130MHz) and 5MHz(2130-2135MHz) can be seen at the experimental site. In order to ensure the accuracy of the experimental results, the base station signal with bandwidth of 5MHz needs to be eliminated from the received signal.

the Flow of Signal Processing



Load Data

```
addpath('data')  
%load data_1.mat  
idx_start_time =1;  
load(sprintf('data/data_%d.mat',idx_start_time))
```

the Sampling rate is 25MHz

(data_1<—>0-0.5s;data_2<—>0.5s-1s;.....);

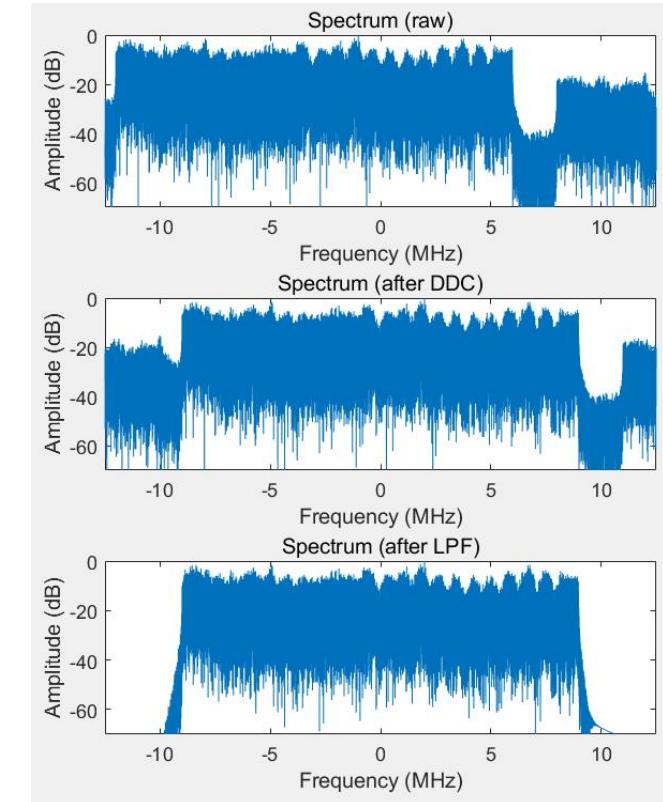
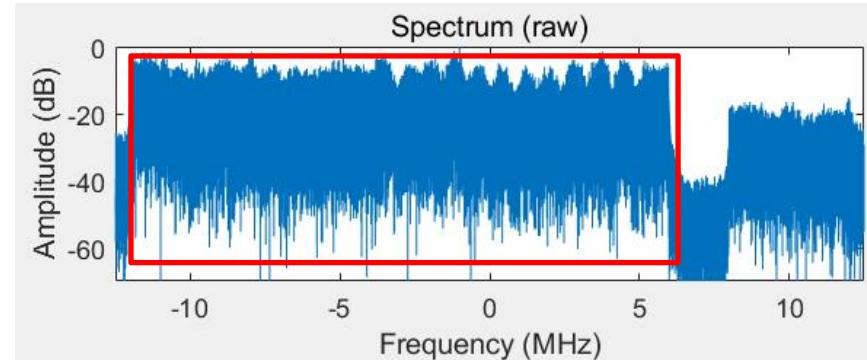
DDC and Lowpass Filter

$f_{ddc} = -3\text{e}6$;
 $\text{bandwidth} = 9\text{e}6$;

Attention: we have two base stations with bandwidth of 20MHz and 5MHz

- 2110-2130MHz
- 2130-2135MHz.

Signal spectrum



Before signal processing, we need to filter out the signal of 2130-2135MHz (5M bandwidth), and reserve the signal of 2110-2130MHz(20M bandwidth). So we need to convert the signal 2110-2130MHz to the frequency centered on 0 Hz. Then filter out the high-frequency signal through the low-pass filter.

Ambiguity Function

Take the segment within $[t, t + T]$ time from the two receiving signals for ambiguity function calculation, calculate the value of ambiguity function by traveling all possible (τ, f_D) , and select $(\hat{\tau}, \hat{f})$ corresponding to the maximum value as the estimate value.

$$Cor(\tau, f_D) = \int_t^{t+T} y_{surv}(t)y_{ref}^*(t - \tau) e^{-j2\pi f_D t} dt \rightarrow Cor(\tau, f_D) = \sum_{n=0}^{N-1} y_{surv}[nT_s]y_{ref}^*[nT_s - \tau] e^{-j2\pi f_D nT_s}$$

- estimate (τ, f_D) :

$$(\hat{\tau}, \hat{f}) = \arg \max_{\tau, f_D} Cor(\tau, f_D)$$

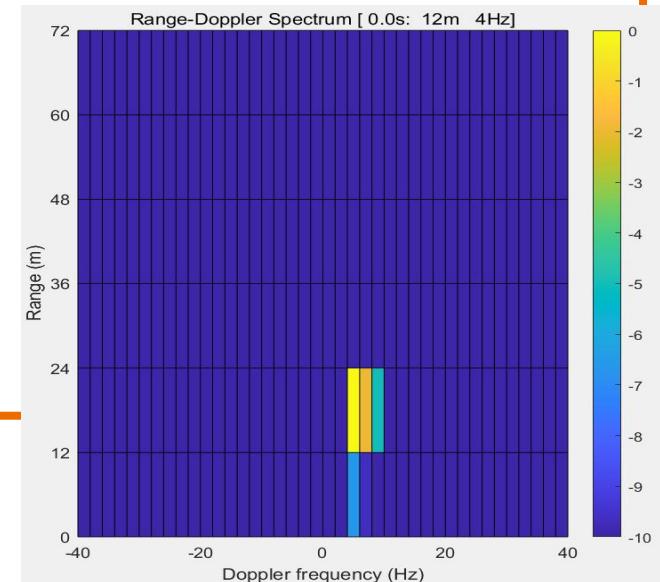
- $\hat{\tau}$ is the estimate of $\tau_s - \tau_r$, \hat{f}_D is the estimate of f .

τ : traversal interval

f_D : traversal interval

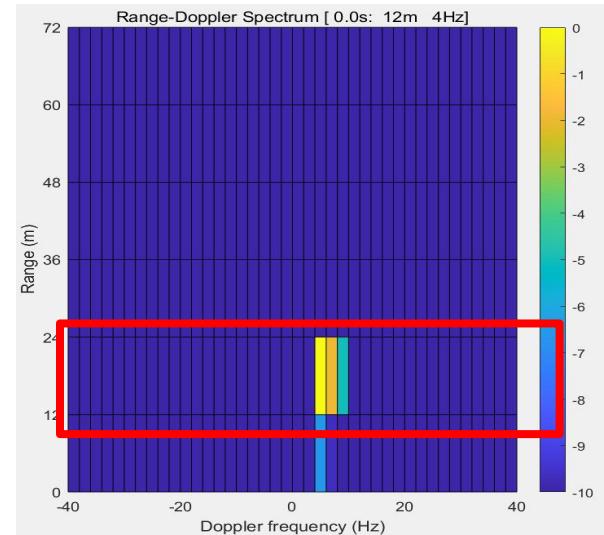


fc is 2123MHz

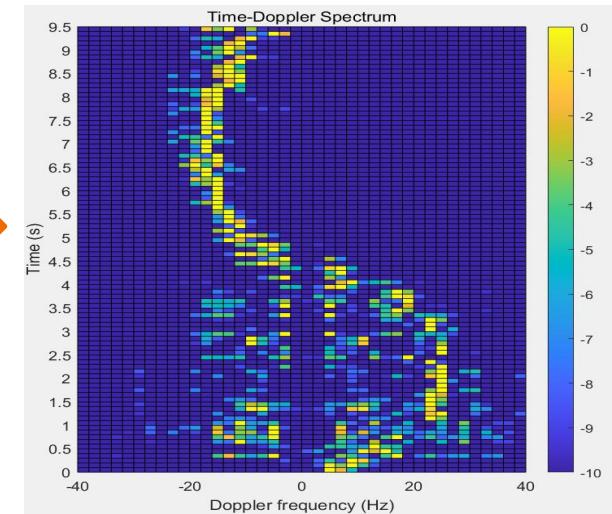


$$Cor(\tau, f_D, t) = \int_t^{t+T} y_{\text{sur}}(t)y_{\text{ref}}^*(t - \tau)e^{-j2\pi f_D t} dt$$

t is a constant

$$Cor(\tau, f_D, t) = \int_t^{t+T} y_{\text{sur}}(t)y_{\text{ref}}^*(t - \tau)e^{-j2\pi f_D t} dt$$

$$g(d, t) = \max_{\tau} |Cor(\tau, f_D, t)|$$

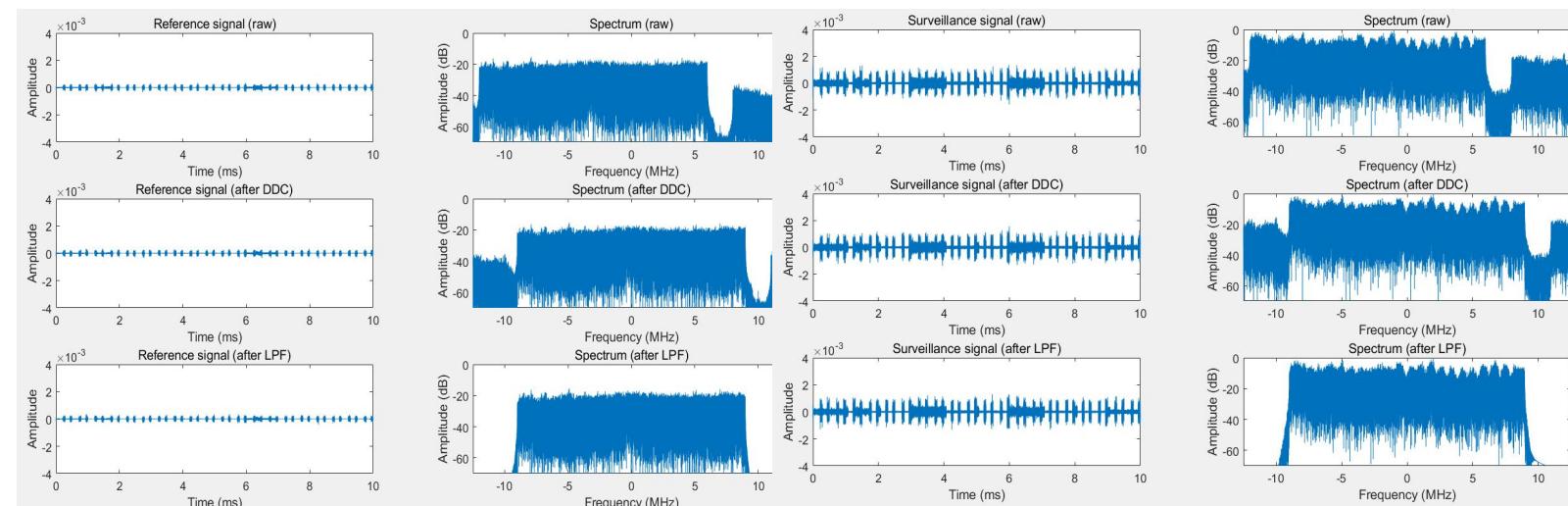
Project Tasks

Task1: For the reference signal and the surveillance signal, draw their time domain waveform and their frequency domain waveform in their following processing process respectively(**0.01s's data is OK**):

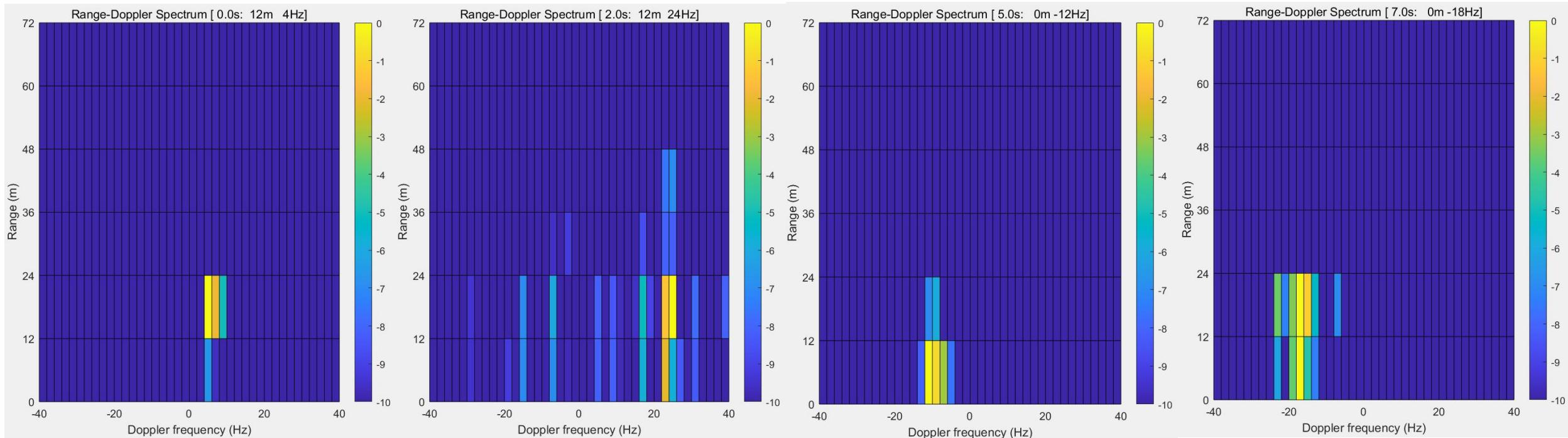
✓ Original(data_1—0-0.5s;data_2—0.5s-1s.....);

✓ After frequency conversion,;

✓ After low-pass filtering;

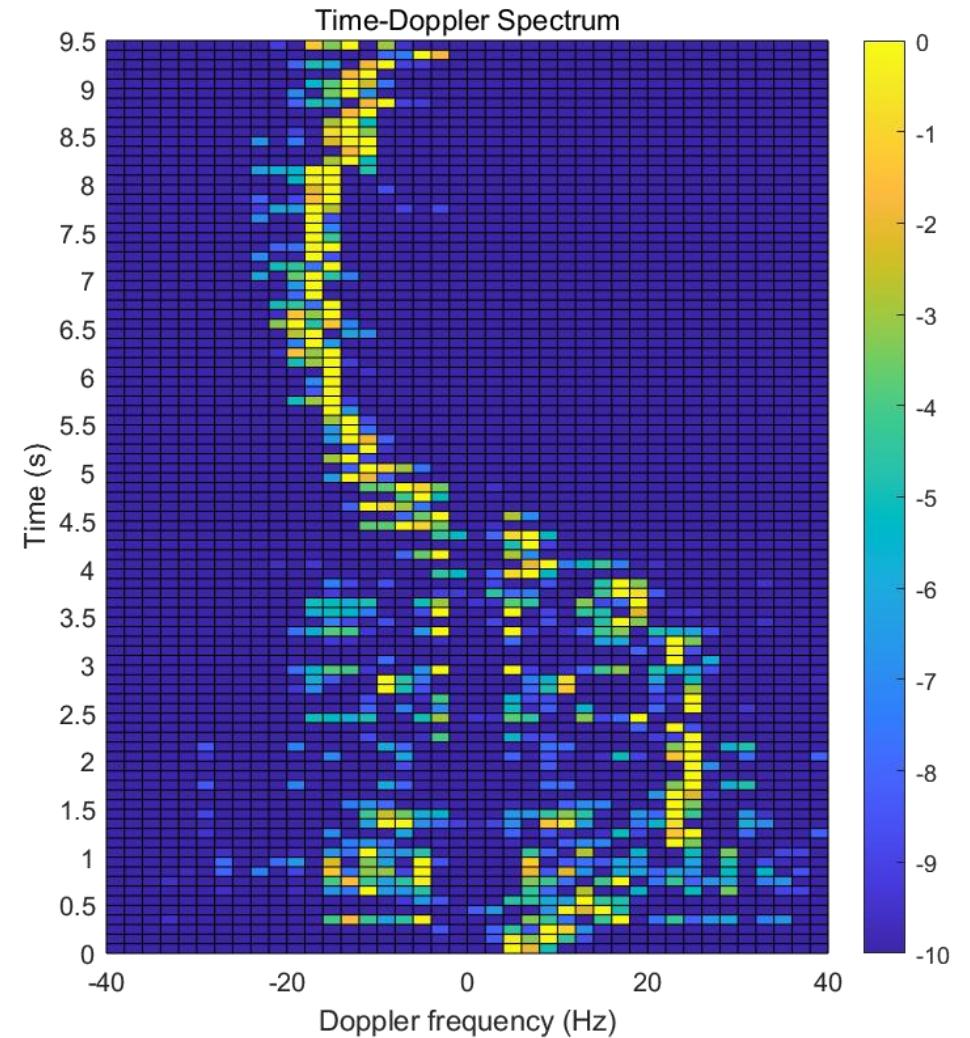
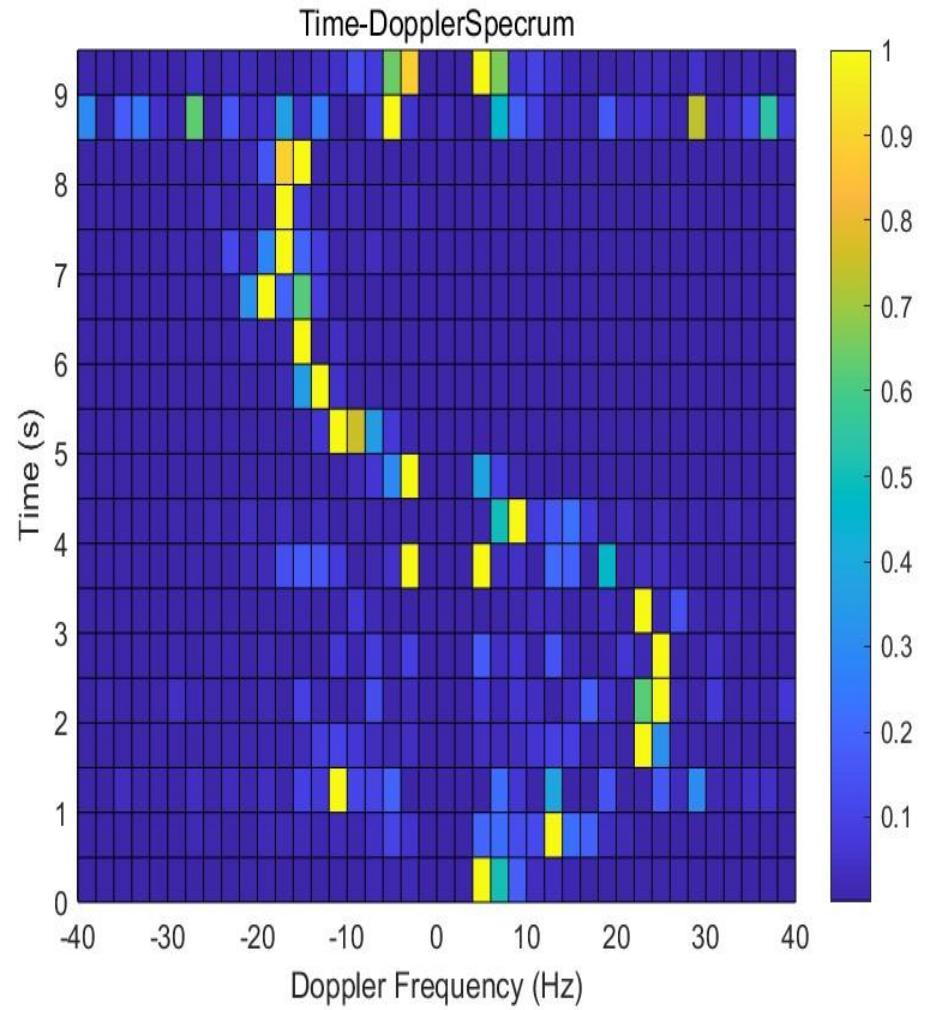


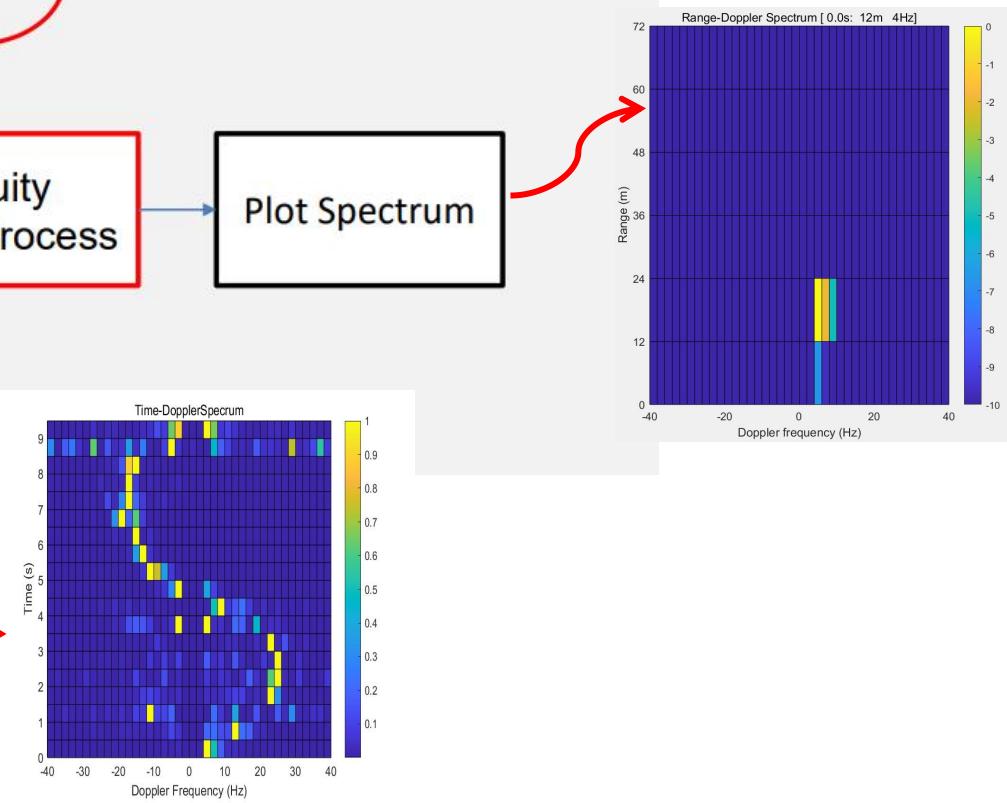
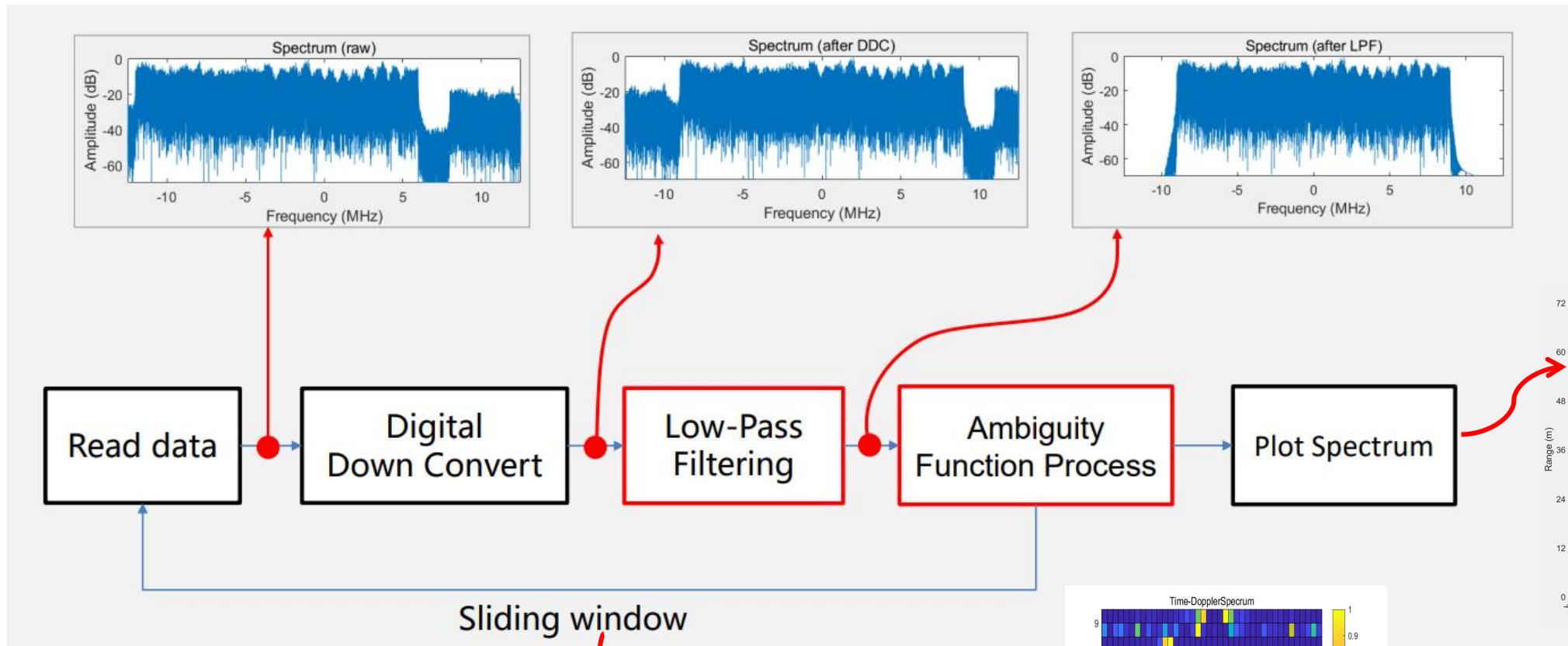
Task2: Draw the Range Doppler spectrum of the signal located at $0 \sim 0.5\text{s}$, $2 \sim 2.5\text{s}$, $5 \sim 5.5\text{s}$ and $7 \sim 7.5\text{s}$ (data is provided)



Don't be afraid, your result may not be the same

Task3: Draw the Time-Doppler spectrum of the signal





Organization

- Each group consists of **three or four** students.
- Each group need present **two Lab projects** (submit reports for both projects):
 - The presentation date is Dec. 28th or 29th for project 1, and Jan. 4th or 5th for Project 2.
- Each presentation is 15 minutes (including Q & A)
 - All team members need to contribute to the presentation.
 - Presentation in English or Chinese.

The presentation should...

- Introduce

- team
- objective of the project
- background review (search more additional information)
- methodology

- Discuss

- what you have learned from this study?
- problems during this project and your solution
- investigation beyond project tasks
- critical thinking

- Present

- relevant data, figure, etc.
- the results for project tasks (e.g., with demo, Figure, etc.)
- interpretation of project findings

- Appendix (if any)
- Team effort (e.g., individual contribution)
- Reference
- Q & A (answer questions raised from audience)

实验报告内容：

Introduction (20%) :

- 1、掌握时域处理方法：（1）数字下变频、（2）数字滤波器设计及使用；（2）互相关计算（模糊函数）
- 2、掌握频率处理方法：（1）分析信号频谱；（2）低通滤波设计及验证；
- 3、基于通信信号的运动检测 技术在生活中的应用

Result and Analysis:

1、完成PPT要求的3个tasks (30%)

注意：（1）低通滤波器系统必须验证；（2）各个节点的时域和频域 图要画出；

2、结果图（时域波形【重要：所有图横坐标以时间秒为单位，纵坐标为幅度，所有坐标轴需要标签】，频谱【重要：所有图横坐标以频率Hz为单位，纵坐标为幅度谱，所有坐标轴需要标签】）(30%)

3、对结果的分析和解释（对结果必须有分析、解释和结论）(10%)

4、加分项：鼓励对项目基础部分进行扩展、研究、讨论，内容自拟，比如：算法的优化，时间多普勒图在时间刻度上的进一步细分，等；加分项需要在Pre中明确讲出加分项占比不超过10%。(10%)

5、总结：在MATLAB仿真中遇到的问题，以及收获的经验。

Score: 自我评分 (0-100, 小组成员分别给出。明确标注报告贡献百分比。未明确标注的，默认贡献均等。)

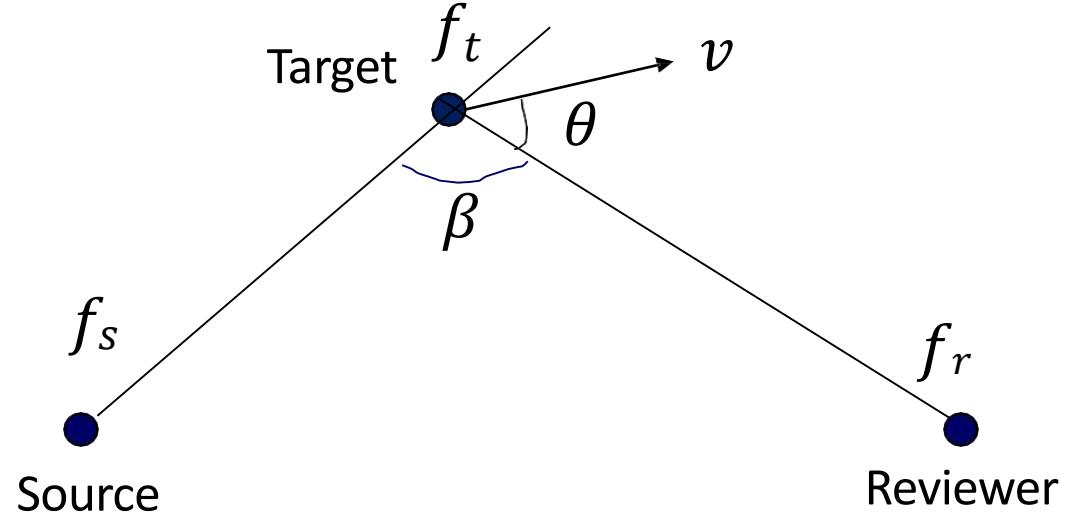
Code: 报告的最后需要附上所有代码。

Questions



About f_D

$$f' = \left(\frac{v \pm v_o}{v \mp v_s} \right) f$$



$$\begin{aligned} f_t &= \frac{c + v \cos(\theta + \beta)}{c} f_s \\ f_r &= \frac{c}{c - v \cos \theta} f_t \\ &= \frac{c + v \cos(\theta + \beta)}{c - v \cos \theta} f_s \end{aligned}$$

$$\begin{aligned} f_D &= f_r - f_s = \frac{v \cos(\theta + \beta) + v \cos \theta}{c - v \cos \theta} f_s = \frac{v f_s}{c} \frac{\cos(\theta + \beta) + \cos \theta}{1 - \frac{v}{c} \cos \theta} \\ &= \frac{v f_s}{c} [\cos(\theta + \beta) + \cos \theta] = \frac{2v}{\lambda_s} \cos \left(\theta + \frac{\beta}{2} \right) \cos \frac{\beta}{2} \end{aligned}$$

$$f_D = \frac{2v}{\lambda_s} \cos\left(\theta + \frac{\beta}{2}\right) \cos\frac{\beta}{2}$$

$\beta \approx 90^\circ, \theta = 0 \text{ or } \pi$

$$f_D = \pm \frac{v}{\lambda_s}$$

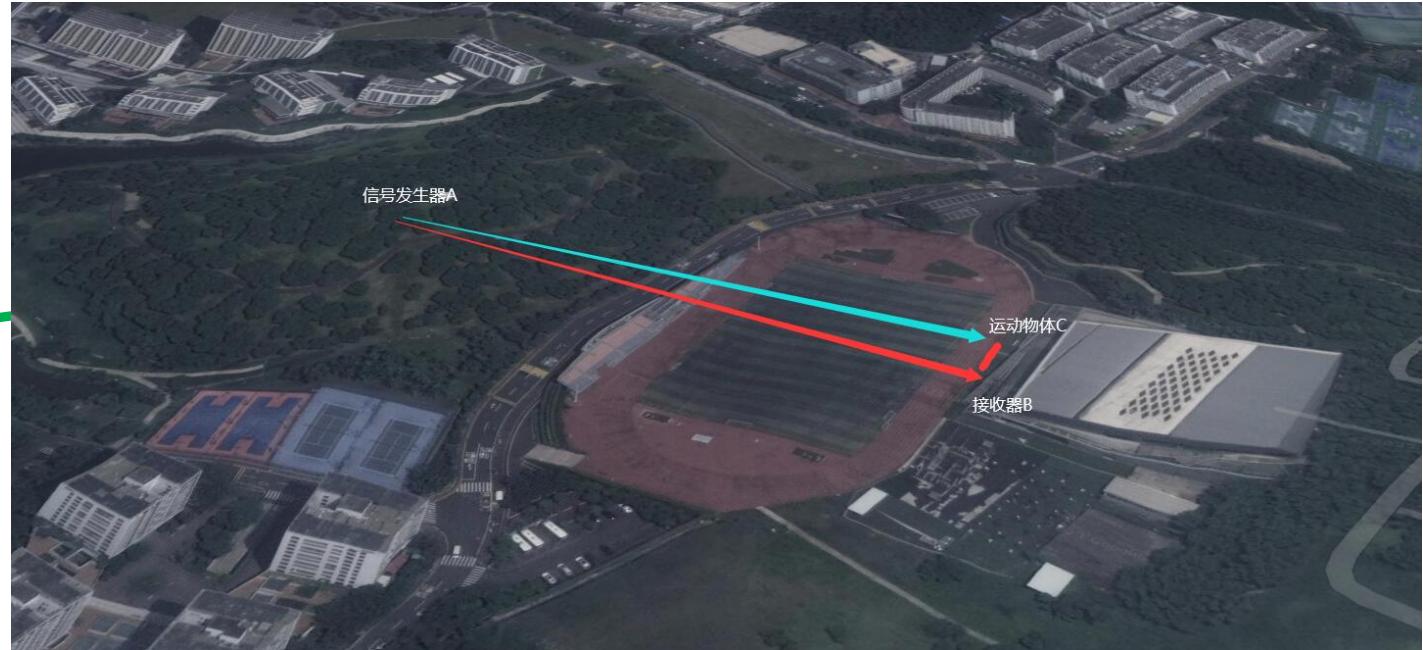


figure--Create figure window

Syntax

```
figure  
figure('PropertyName',propertyvalue,...)  
figure(h)  
h = figure(...)
```

Properties

For a list of properties, see [Figure Properties](#).

Description

`figure` creates a new `figure` window using default property values. This new `figure` window becomes the current `figure`, and it displays on top of all other figures on the screen. The title of the `figure` is an integer value that is not already used by an existing `figure`. MATLAB® saves this integer value in the `figure`'s `Number` property.

`figure('PropertyName',propertyvalue,...)` creates a new `figure` window using specific property values. For a list of available properties, see [Figure Properties](#). MATLAB uses default values for any properties that you do not explicitly define as arguments.

`figure(h)` does one of the following:

- If `h` is the handle or the `Number` property value of an existing `figure`, then `figure(h)` makes that existing `figure` the current `figure`, makes it visible, and moves it on top of all other figures on the screen. The current `figure` is the target for graphics output.
- If `h` is not the handle and is not the `Number` property value of an existing `figure`, but is an integer, then `figure(h)` creates a `figure` object and assigns its `Number` property the value `h`.
- If `h` is not the handle to a `figure` and is not a positive integer, then MATLAB returns an error.

`h = figure(...)` returns the handle to the `figure` object.

Figure Properties

Control appearance and behavior of figure window

[expand all in page](#)

Figures are windows that contain graphics or user interface components. Figure properties control the appearance and behavior of a particular instance of a figure. To modify aspects of a figure, change property values.

Starting in R2014b, you can use dot notation to query and set properties.

```
fig = figure;  
u = fig.Units;  
fig.Units = 'inches';
```

If you are using an earlier release, use the [get](#) and [set](#) functions instead.

Figure Appearance

[expand all](#)

▼ **Color — Figure window background color**
RGB triplet | short name | long name | 'none'

▼ **DockControls — Interactive figure docking**
'on' (default) | 'off'

▼ **MenuBar — Figure menu bar display**
'figure' (default) | 'none'

get--Query graphics object properties

gcf--Current figure handle

gca--Current axes handle

gco--Handle of current object

gcp--Get current parallel pool

set--Set graphics object properties

saveas--Save figure to specific file format

Syntax

```
saveas(fig,filename)  
saveas(fig,filename,formattype)
```

Description

`saveas(fig,filename)` saves the figure or Simulink® block diagram specified by `fig` to file `filename`. Specify the file name as a character vector that includes a file extension, for example, '`myplot.jpg`'. The file extension defines the file format. If you do not specify an extension, then `saveas` saves the figure to a FIG-file. To save the current figure, specify `fig` as `gcf`.

`saveas(fig,filename,formattype)` creates the file using the specified file format, `formattype`. If you do not specify a file extension in the file name, for example, '`myplot`', then the standard extension corresponding to the specified format automatically appends to the file name. If you specify a file extension, it does not have to match the format. `saveas` uses `formattype` for the format, but saves the file with the specified extension. Thus, the file extension might not match the actual format used.

print--Print figure or save to specific file format
savefig--Save figure and contents to FIG-file

```
clc,clear,close all
```

```
scrsz =get(0,'ScreenSize')  
figure1=figure('Position',[0 30 scrsz(3) scrsz(4)-95])
```

```
%生成数据
```

```
x=rand(100,1)
```

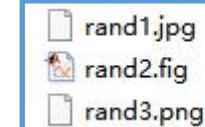
```
%画图保存
```

```
plot(x)
```

```
saveas(figure1,'rand1','jpg')
```

```
savefig(figure1,'rand2')
```

```
print('rand3','-dpng')
```



sprintf--Format data into string

Syntax

```
str = sprintf(formatSpec,A1,...,An)  
[str,errmsg] = sprintf(formatSpec,A1,...,An)
```

Description

`str = sprintf(formatSpec,A1,...,An)` formats the data in arrays A1,...,An according to formatSpec in column order, and returns the results to str.

`[str,errmsg] = sprintf(formatSpec,A1,...,An)` returns an error message as a character vector when the operation is unsuccessful. Otherwise, errmsg is empty.

imagesc--Display image with scaled colors

Syntax

```
imagesc(C)
```

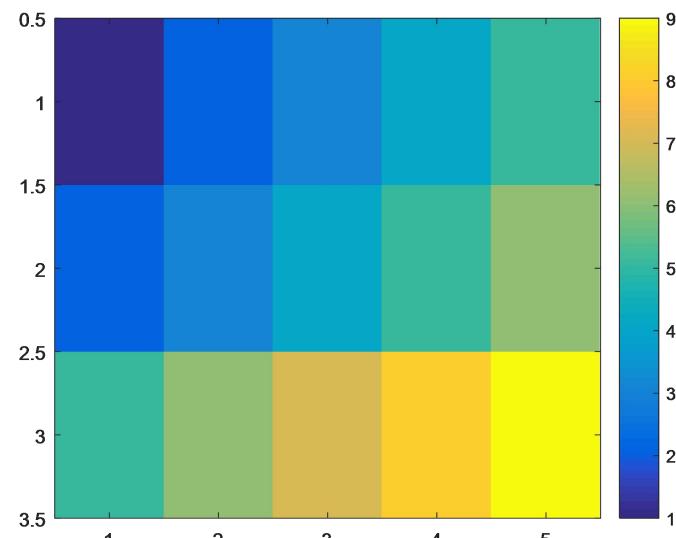
example

```
imagesc(x,y,C)
```

example

`imagesc(x,y,C)` specifies the image location. Use `x` and `y` to specify the locations of the corners corresponding to $C(1,1)$ and $C(m,n)$. To specify both corners, set `x` and `y` as two-element vectors. To specify the first corner and let `imagesc` determine the other, set `x` and `y` as scalar values. The image is stretched and oriented as applicable.

```
close all  
x=[1 2 3 4 5];  
y=[1 2 3 ];  
C=[1 2 3 4 5;2 3 4 5 6;5 6 7 8 9;]  
imagesc(x,y,C)  
colorbar
```



meshgrid--2-D and 3-D grids

Syntax

```
[X,Y] = meshgrid(x,y)  
[X,Y] = meshgrid(x)  
  
[X,Y,Z] = meshgrid(x,y,z)  
[X,Y,Z] = meshgrid(x)
```

Description

`[X,Y] = meshgrid(x,y)` returns 2-D grid coordinates based on the coordinates contained in vectors `x` and `y`. `X` is a matrix where each row is a copy of `x`, and `Y` is a matrix where each column is a copy of `y`. The grid represented by the coordinates `X` and `Y` has `length(y)` rows and `length(x)` columns.

`[X,Y] = meshgrid(x)` is the same as `[X,Y] = meshgrid(x,x)`, returning square grid coordinates with grid size `length(x)`-by-`length(x)`.

`[X,Y,Z] = meshgrid(x,y,z)` returns 3-D grid coordinates defined by the vectors `x`, `y`, and `z`. The grid represented by `X`, `Y`, and `Z` has size `length(y)`-by-`length(x)`-by-`length(z)`.

`[X,Y,Z] = meshgrid(x)` is the same as `[X,Y,Z] = meshgrid(x,x,x)`, returning 3-D grid coordinates with grid size `length(x)`-by-`length(x)`-by-`length(x)`.

surf--3-D shaded surface plot

Syntax

```
surf(z)
surf(z,c)
surf(x,y,z)
surf(x,y,z,c)
surf(...,'PropertyName',PropertyValue)
surf(ax,...)
h = surf(...)
```

Description

`surf(z)` creates a three-dimensional shaded surface from the z components in matrix z, using $x = 1:n$ and $y = 1:m$, where $[m,n] = \text{size}(z)$. The height, z, is a single-valued function defined over a geometrically rectangular grid. z specifies the color data, as well as surface height, so color is proportional to surface height. The values in z can be numeric or datetime or duration values.

`surf(z,c)` plots the height of z, a single-valued function defined over a geometrically rectangular grid, and uses matrix c, assumed to be the same size as z, to color the surface. See [Coloring Mesh and Surface Plots](#) for information on defining c.

`surf(x,y,z)` uses z for the color data and surface height. x and y are vectors or matrices defining the x and y components of a surface. If x and y are vectors, $\text{length}(x) = n$ and $\text{length}(y) = m$, where $[m,n] = \text{size}(z)$. In this case, the vertices of the surface faces are $(X(j), Y(i), Z(i,j))$ triples. To create X and Y matrices for arbitrary domains, use the `meshgrid` function. The values in x, y, or z can be numeric or datetime or duration values.

`surf(x,y,z,c)` uses c to define color. MATLAB® performs a linear transformation on this data to obtain colors from the current colormap.

`surf(...,'PropertyName',PropertyValue)` specifies surface properties along with the data. For a list of properties, see [Chart Surface Properties](#).

`surf(ax,...)` plots into the axes ax instead of the current axes (`gca`).

`h = surf(...)` returns a handle to a chart surface graphics object.

view--Viewpoint specification

Syntax

```
view(az,el)
view([az,el])
view([x,y,z])
view(2)
view(3)
view(ax,...)
[az,el] = view
```

Description

The position of the viewer (the viewpoint) determines the orientation of the axes. You specify the viewpoint in terms of azimuth and elevation in three-dimensional space.

`view(az,el)` and `view([az,el])` set the viewing angle for a three-dimensional plot. The azimuth, `az`, is the horizontal rotation angle measured in degrees from the negative `y`-axis. Positive values indicate counterclockwise rotation of the viewpoint. `el` is the vertical elevation angle measured in degrees. Positive values of elevation correspond to moving above the object; negative values correspond to moving below it.

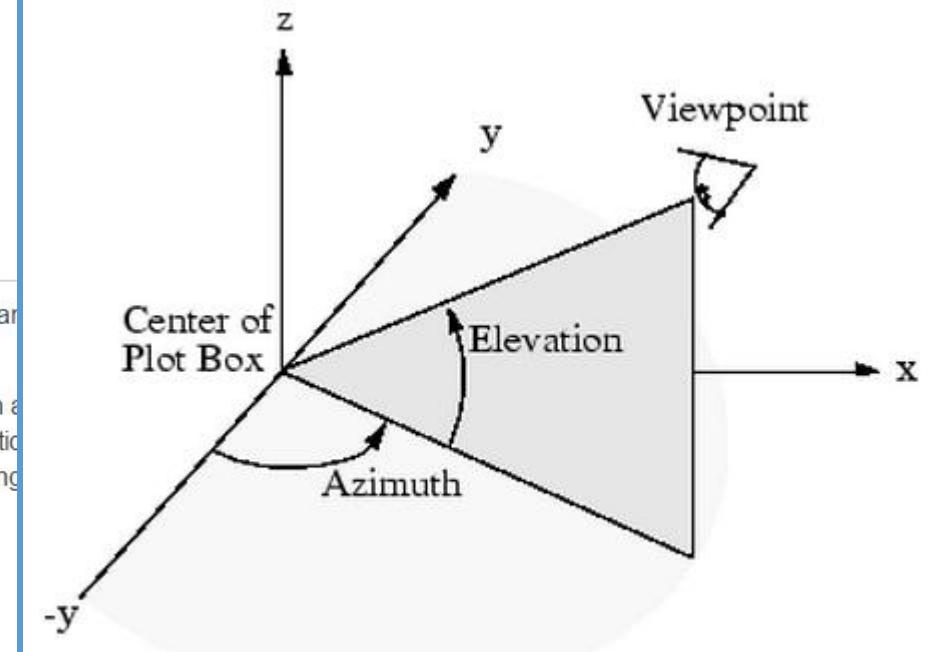
`view([x,y,z])` sets the `view` direction to the Cartesian coordinates `x`, `y`, and `z`. The magnitude of (x,y,z) is ignored.

`view(2)` sets the default two-dimensional `view`, `az = 0, el = 90`.

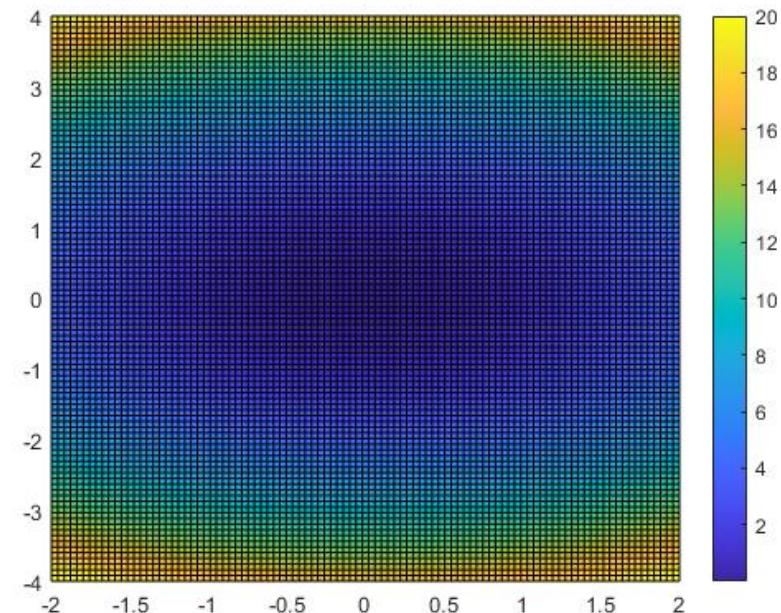
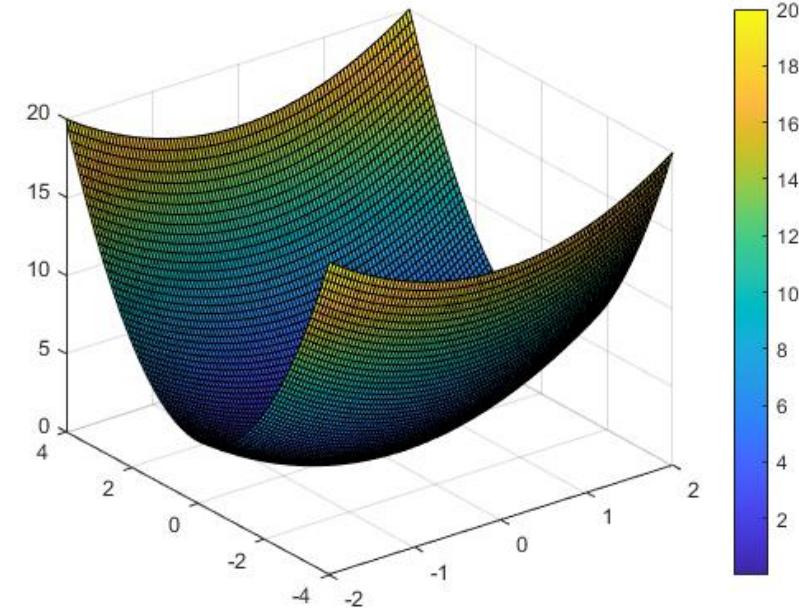
`view(3)` sets the default three-dimensional `view`, `az = -37.5, el = 30`.

`view(ax,...)` uses axes `ax` instead of the current axes.

`[az,el] = view` returns the current azimuth and elevation.



```
clc,clear,close all  
  
x = linspace(-2,2,100)  
y = linspace(-4,4,100)  
  
[X,Y] = meshgrid(x,y)  
  
% grid.  
  
Z= X.^2+Y.^2;  
  
figure  
surf(X,Y,Z),colorbar  
[az el]=view  
  
figure  
surf(X,Y,Z),colorbar  
view(0,90)
```

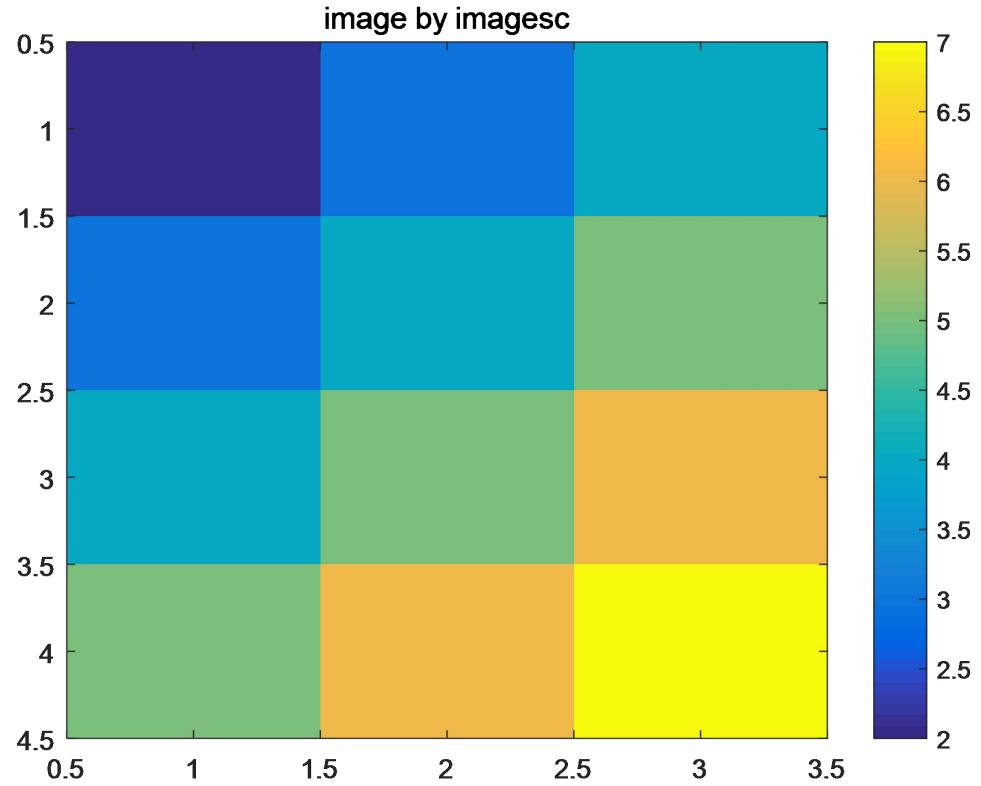


imagesc VS surf

```
clc,clear,close all
```

```
m=[1 2 3]
```

```
n=[1 2 3 4]
```



```
[X,Y]=meshgrid(m,n)
```

```
figure
```

```
surf(X,V,V)
```

