1. 介绍

使用数据采集卡 DCA1000EVM 可以将所有 LVDS 数据均按原数据格式捕获并通过以太网接口传输出来。mmWave Studio 是一款独立的 Windows GUI，它具有配置和控制 TI 毫米波传感器模块以及收集 ADC 数据以进行离线分析的功能。

DCA1000+mmWave Studio 软件进行毫米波雷达数据采集的框图如图 1 所示。

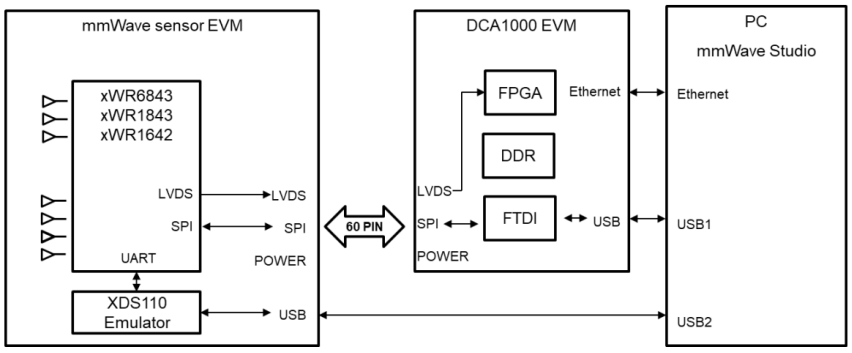

图 1

使用 DCA1000+mmWave Studio 软件进行数据采集的软硬件要求如下：

硬件：mmWave radar sensor EVM + DCA1000 + PC

软件：mmWave studio

Radar 硬件所需预留接口：LVDS（ADC 数据传输）+ SPI（RF 参数配置）+ UART（固件加载）+ SOP（SOC 启动模式配置）。

2. mmWave studio 的使用

(1) 流程

详见 DCA1000EVM Quick Start Guide、mmWave Sensor Raw Data Capture Using the DCA1000 Board and mmWave Studio 或 mmwave_studio_user_guide。

(2) 手动配置

StaticConfig 里 LP ADC Mode 配置为 Low Power ADC，SensorConfig 注意采样率不要过大（默认 1e6），其他都使用默认配置就可以开始采集数据。

手势数据采集要求的雷达配置如图 2 所示。此外，雷达发射通道数为 2，接收通道数为 4。

名称	数值
载频	77 GHz
带宽	3.072 GHz
时宽	102.4us
PRF	2000Hz
调频斜率	30MHz/us
距离采样点	512
ADC 采样率	5MHz

图 2

手动配置 Connection、StaticConfig、DataConfig、SensorConfig 如下图 3、4、5、6 所示。需要注意的是，chirpcfg 中，start chirp for cfg 和 end chirp for cfg 的意义是希望设置从 start index 开始到 end index 结束都是这个 chirp 配置。

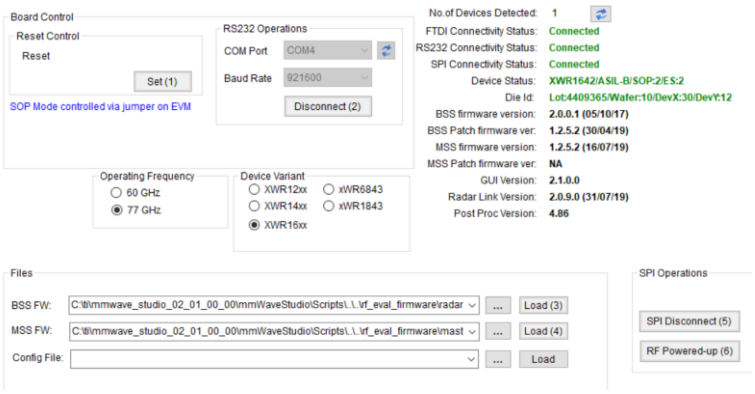
Board Control
Reset Control
Reset
Set (1)
SOP Mode controlled via jumper on EVM
RS232 Operations
COM Port
COM4
Baud Rate
921600
Disconnect (2)
No.of Devices Detected: 1
FTDI Connectivity Status: Connected
RS232 Connectivity Status: Connected
SPI Connectivity Status: Connected
Device Status: XWR1642/ASIL-B/SOP:2/ES:2
Die Id: Lot:4409365/Wafer:10/DevX:30/DevY:12
BSS firmware version: 2.0.0.1 (05/10/17)
BSS Patch firmware ver: 1.2.5.2 (30/04/19)
MSS firmware version: 1.2.5.2 (16/07/19)
MSS Patch firmware ver: NA
GUI Version: 2.1.0.0
Radar Link Version: 2.0.9.0 (31/07/19)
Post Proc Version: 4.86
Operating Frequency
60 GHz
77 GHz
Device Variant
XWR12xx
xWR6843
XWR14xx
xWR1843
XWR16xx
Files
BSS FW:
C:\ti\mmwave_studio_02_01_00_00\mmWaveStudio\Scripts\..\..\rf_eval_firmware\radar
Load (3)
MSS FW:
C:\ti\mmwave_studio_02_01_00_00\mmWaveStudio\Scripts\..\..\rf_eval_firmware\mast
Load (4)
Config File:
Load
SPI Operations
SPI Disconnect (5)
RF Powered-up (6)

图 3

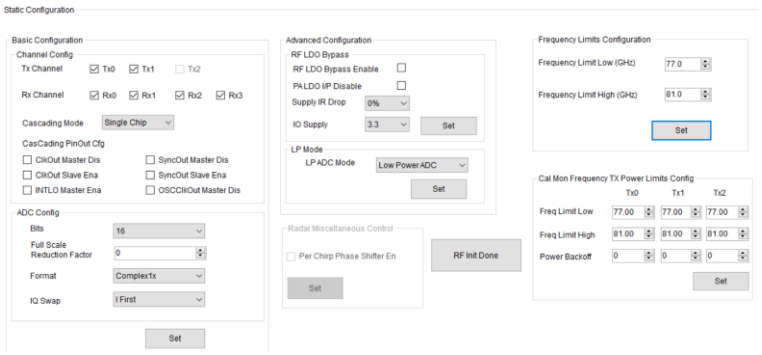
Static Configuration
Basic Configuration
Channel Config
Tx Channel
Rx Channel
Cascading Mode
Single Chip
CasCading PinOut Cfg
ADC Config
Bits
Full Scale Reduction Factor
Format
Complex1x
IQ Swap
I First
Set
Advanced Configuration
RF LDO Bypass
RF LDO Bypass Enable
PALDO I/P Disable
Supply IR Drop
IO Supply
LP Mode
LP ADC Mode
Low Power ADC
Radar Miscellaneous Control
Per Chirp Phase Shifter En
RF Init Done
Frequency Limits Configuration
Frequency Limit Low (GHz)
77.0
Frequency Limit High (GHz)
81.0
Cal Mon Frequency TX Power Limits Config
Freq Limit Low
Freq Limit High
Power Backoff

图 4

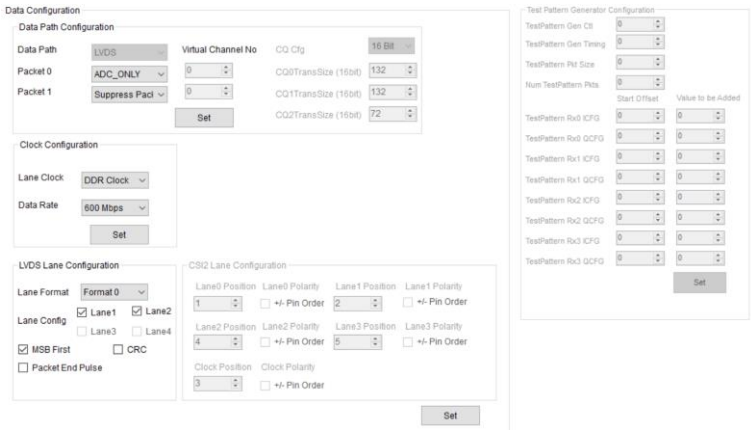
Data Configuration
Data Path Configuration
Data Path
LVDS
Virtual Channel No
Packet 0
ADC_ONLY
Packet 1
Suppress Pacl
Set
Clock Configuration
Lane Clock
DDR Clock
Data Rate
600 Mbps
LVDS Lane Configuration
Lane Format
Format 0
Lane Config
Lane1
Lane2
Lane3
Lane4
MSB First
CRC
Packet End Pulse
CSI2 Lane Configuration
Test Pattern Generator Configuration

图 5

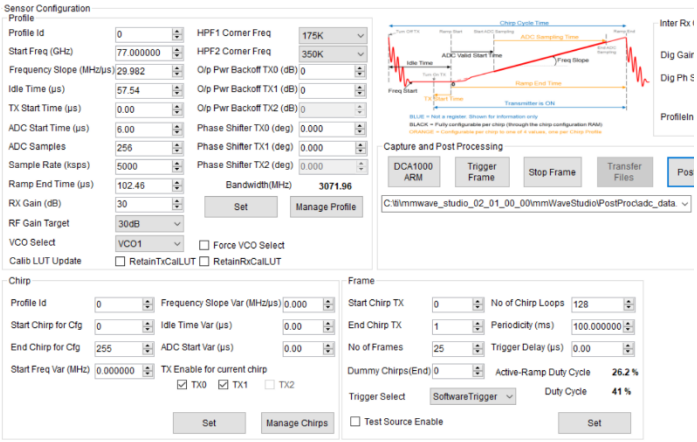
Sensor Configuration
Profile
Profile Id
Start Freq (GHz)
77.000000
Frequency Slope (MHz/µs)
29.982
Idle Time (µs)
57.54
TX Start Time (µs)
0.00
ADC Start Time (µs)
6.00
ADC Samples
256
Sample Rate (ksps)
5000
Ramp End Time (µs)
102.46
RX Gain (dB)
30
RF Gain Target
30dB
VCO Select
VCO1
Calib LUT Update
HPF1 Corner Freq
175K
HPF2 Corner Freq
350K
Bandwidth(MHz)
3071.96
Set
Manage Profile
Force VCO Select
RetainTxCalLUT
RetainRxCalLUT
Capture and Post Processing
DCA1000 ARM
Trigger Frame
Stop Frame
Transfer Files
Chirp
Profile Id
Start Chirp for Cfg
End Chirp for Cfg
Start Freq Var (MHz)
Frequency Slope Var (MHz/µs)
Idle Time Var (µs)
ADC Start Var (µs)
TX Enable for current chirp
TX0
TX1
TX2
Set
Manage Chirps
Frame
Start Chirp TX
End Chirp TX
No of Frames
Dummy Chirps(End)
Trigger Select
SoftwareTrigger
No of Chirp Loops
Periodicity (ms)
Trigger Delay (µs)
Active-Ramp Duty Cycle
26.2 %
Duty Cycle
41 %
Test Source Enable
Set

图 6

(3) 相关问题

使用 DCA1000+mmWave Studio 软件进行数据采集，在连接过程中出现的问题，可以参考 DCA1000 Debugging Handbook。

另外，记录一个较为在意的问题。当使用 Ti 的毫米波评估板采集数据时，连接问题多卡在最后一步“FPGA 的版本识别”。

不能在上位机识别 FPGA 版本是以太网络的问题，解决方法如下：

1) 确认本地连接的以太网的静态 IP 是否正确。
2) 防火墙会对未知的网络阻止数据传输，确认防火墙已关闭。
3) 网口必须是千兆网口，百兆网口不满足 Ti 规定的传输速率。
4) 尝试刷新 FPGA 固件，详见 DCA1000EVM Data Capture Card 第九章 FPGA Programming。
5) 硬件问题，如网线端口坏了或网线松动等。

3. 从 adc_data.bin 文件读取数据

捕获的原始数据由数据采集卡通过以太网接口传输到 PC，并由 mmWave Studio 收集后保存在 ti\mmwave_studio_<ver>\mmWaveStudio\PostProc 下的 adc_data.bin 文件中。其中，数据的格式如图 7 所示。

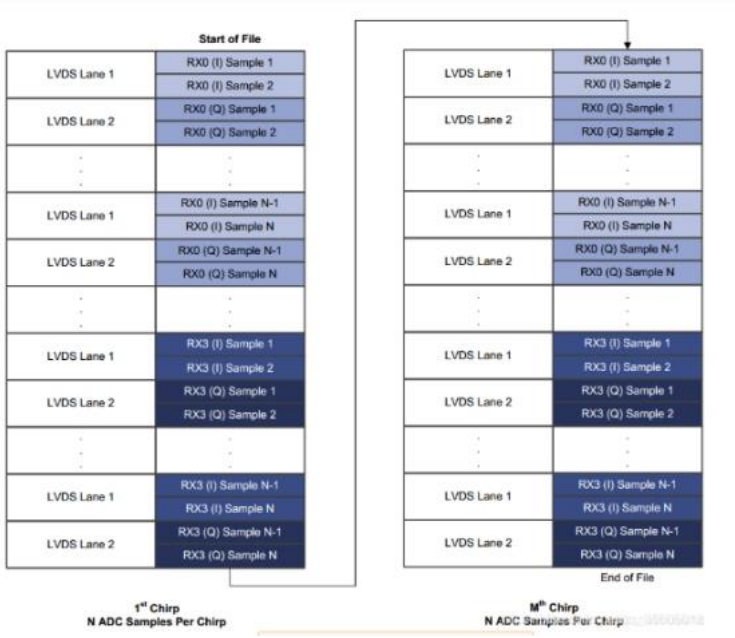

图 7

数据的组织格式为 Sample->RX->Chirp，存放方式为小端模式。每个数据采样点分为实部（I 路）和虚部（Q 路），大小均为 2 字节。基于上述特点，给出回波数据读取的代码，如图 8 所示。

```
fname='adc_data.bin';
fid = fopen(fname,'rb');
bufferSize=NSample*NChirp*Rx_Number*Tx_Number*2;
%16bits，复数形式(I/Q两路)，4RX,1TX,有符号16bit，小端模式
sdata = fread(fid,frame_num*bufferSize,'int16');
fclose(fid);
```

图 8

由于 TX 轮流发射 chirp，因此总的 chirp 数为 NChirp(每帧每个 TX 天线发射的 chirp 数)*Tx_Number(TX 天线数)，每一帧的大小为 bufferSize=NSample(每个 chirp 的采样点数)*RX_Number(RX 天线数)*总 chirp 数，从文件中读取的数据的大小为 frame_num(帧数)*bufferSize，读取格式为 16 位有符号整型数。

4. 处理思路

对获取的原始数据进行处理并获得手势信息，主要有以下两种思路。

第一种思路，将每一帧的数据排列成三维矩阵，并在相应维依次进行距离 FFT、速度 FFT 和角度 FFT，再对多个帧的数据进行汇总，得到手势信息，其对应的处理流程为后文的 5、6、7 部分。

第二种思路，将多帧的原始数据汇总，排列成三维矩阵（单次处理的 chirp 数大大增加，本文中为 8000 个 chirp），并对其进行距离 FFT、时频分析、music 处理，从而得到手势信息，其对应的处理流程为后文的 8、9、10 部分。

基于第一种思路，使用 AWR1642 获取到的手势信息图，除上拉、下推和上下轻拍外皆不够理想，这可能与雷达本身有关，也可能与配置或实验环境有关。此外，由于现实生活中信号一般呈现非平稳，平稳信号多为理想状况，因此 FFT 不足以对手势回波信号进行分析。

在分别基于两种思路进行过若干次实验后，本文最终选择第二种思路。之所以没有删除第一种思路的相关内容，主要是为了对工作内容进行记录，因此在阅读本文时，这一部分的内容可以跳过。

5. 单帧数据处理（思路一）

对于获取到的数据，每次从中取出一帧的数据进行处理。处理函数 Three_dfft 的输入为 1*bufferSize 的向量 fdata，输出为列向量 fft1d、fft2d 和 fft3d。

(1) 数据拆分、组合

第一步是将虚部和实部数据重组得到复信号。值得注意的是，LVDS Lane 1 只存放 I 路数据，LVDS Lane 2 只存放 Q 路数据，因此处理时每次重组 4 个数据点得到 2 个采样点的复信号。相关代码如图 9 所示。

```
fileSize = size(fdata, 1);
lvds_data = zeros(1, fileSize/2);
count = 1;
for i=1:4:fileSize-5
    lvds_data(1,count) = fdata(i) + 1i*fdata(i+2);
    lvds_data(1,count+1) = fdata(i+1)+1i*fdata(i+3);
    count = count + 2;
end
```

图 9

第二步是将复信号数据 lvds_data 排列成 3 维矩阵 ADC_Data(速度维、距离维和天线维)，根据前文对数据组织格式的讨论，容易得知 lvds_data 的数据格式如图 10 所示。

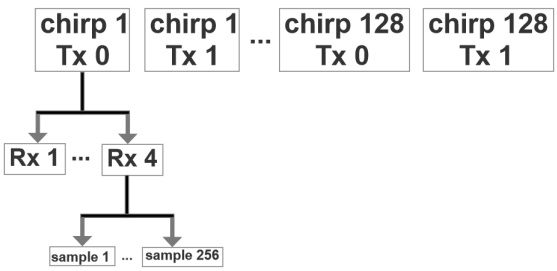

图 10

每个 chirp 有 Rx_Number 个通道，处理时不妨将多个 Tx 天线发射的第 k 个 chirp 的数据进行合并，从原来的 Rx_Number 个通道变为 Rx_Number*Tx_Number 个通道。那么，每个通道的数据大小为 xSize=Range_Number，每个 chirp 的数据大小为 chirpSize=xsize*通道数。

根据以上讨论，可以得到排列数据的代码，如图 11 所示。

```
ADC_Data=zeros(Doppler_Number,Range_Number,Rx_Number*Tx_Number); %建立计算存储数据的空矩阵

xSize=Range_Number;
chirpSize=xSize*Rx_Number*Tx_Number;
for i=1:Doppler_Number
    for k=1:Rx_Number*Tx_Number
        ADC_Data(i,:,k) = lvds_data((i-1)*chirpSize+(k-1)*xSize+1:(i-1)*chirpSize+k*xSize);
    end
end
```

图 11

(2) 距离 FFT

对 ADC_Data 的单个脉冲采样数据（距离维）进行 1D FFT 得到矩阵 range_profile，对各脉冲信息整合，得到整体的距离变化信息。这里加窗主要是为了使时域信号更好地满足 FFT 处理的周期性要求，减少信号的泄漏。

距离 FFT 的代码图 12 所示。

```
range_win = hamming(Range_Number).';   %加海明窗
range_profile = zeros(Doppler_Number,Range_Number,Rx_Number*Tx_Number);
for k=1:Rx_Number*Tx_Number
   for m=1:Doppler_Number
      temp=ADC_Data(m,:,k).*range_win;     %加窗函数
      temp_fft=fft(temp,Range_Number);     %对每个chirp做FFT
      range_profile(m,:,k)=temp_fft;
   end
end
```

图 12

(3) 静态杂波滤除（相量均值相消法）

静态杂波滤除可采用相量均值相消法。其原理为，对所有接收脉冲求平均得出参考接收脉冲，接着利用每一束接收脉冲减去参考接收脉冲就可以得到目标回波信号，参考接收脉冲的表达式为

$$C[m]=\frac{1}{N}\sum_{i=1}^{N}R[m,i]$$

其中，m 为距离维采样点，i 为速度维采样点，相量均值相消算法的公式为 R[m,n]=R[m,n]-C[m]。

对 range_profile 进行静态杂波滤除，并将滤除结果 fft1d_sub 赋值给 range_profile 的代码如图 13 所示。

```
fft1d_sub =zeros(Doppler_Number,Range_Number,Tx_Number*Rx_Number);
for n=1:Tx_Number*Rx_Number
    avg = sum(range_profile(:,:,n))/Doppler_Number;
    for m=1:Doppler_Number
        fft1d_sub(m,:,n) = range_profile(m,:,n)-avg;
    end
end
range_profile =fft1d_sub;
```

图 13

(4) 速度 FFT

对距离 FFT 的结果 range_profile（经静态杂波滤除）在速度维作 1D FFT 得到矩阵 speed_profile。同样需要加窗。

速度 FFT 的代码如图 14 所示。

```
doppler_win = hamming(Doppler_Number);                    %加海明窗
speed_profile = zeros(Doppler_Number,Range_Number,Rx_Number*Tx_Number);
for k=1:Rx_Number*Tx_Number
    for n=1:Range_Number
      temp=range_profile(:,n,k).*doppler_win;             %加窗函数
      temp_fft=fftshift(fft(temp,Doppler_Number));        %对rangeFFT结果进行FFT
      speed_profile(:,n,k)=temp_fft;
    end
end
```

图 14

(5) 直流分量抑制

基于 77GHz/79GHz FMCW 体制的毫米波雷达，在 RD(Ranger-Doppler)阈检测时，我们会遇到一个直流分量的问题，进而导致近距离（2~3 个采样点）无法处理，从而造成一个较大的盲区。

这里采取的方法是将第一个采样点的数据置 0，代码如图 15 所示。

```
speed_profile(:,1,:)=0;
```

图 15

(6) 角度 FFT

对速度 FFT 结果 speed_profile 在天线维作 1D FFT 得到矩阵 angle_profile。设定 Q=180 返回 Q 点 DFT。

角度 FFT 的代码如图 16 所示。

```
angle_profile = zeros(Doppler_Number,Range_Number,Q);
for n=1:Range_Number   %range
    for m=1:Doppler_Number   %chirp
      temp=speed_profile(m,n,:);
      temp_fft=fftshift(fft(temp,Q));     %对2D FFT结果进行Q点FFT
      angle_profile(m,n,:)=temp_fft;
    end
end
```

图 16

(7) 单帧手势信息的获取

距离手势信息 fft1d 的获取步骤如下：

1) 对一帧信号内的 128 个 chirp 信号分别做一维 FFT 得到 range_profile；

2) 把得到的 128 个频谱结果进行相加并求平均，将结果作为该帧信号的距离信息；

速度手势信息 fft2d 获取步骤如下：

1) 对一帧信号做 2D-FFT 运算，得到距离-多普勒谱图 speed_profile；

2) 依据求得的 fft1d，找到该帧对应的距离信息 r（峰值），将此距离信息 r 对应到距离-多普勒谱图中，提取出距离 r 处对应的速度列向量，作为该帧信号的速度信息。

角度手势信息 fft3d 获取步骤如下：

1）对一帧信号做 3D-FFT 运算，得到 angle_profile；

2）依旧求得的 fft1d 和 fft2d，找到该帧对应的距离信息 r（峰值）和速度信息 v（峰值），在 angle_profile 中提取出对应的角度列向量，作为该帧信号的角度信息。

单帧手势信息的获取代码图 17 所示。

```
fft1d=sum(range_profile(:,:,1)).'/Doppler_Number;
[~,col]=max(fft1d);
fft2d=speed_profile(:,col,1);
[~,row]=max(fft1d);
fft3d=angle_profile(row,col,:);
```

图 17

至此，获得单帧数据处理的结果 fft1d、fft2d 和 fft3d。

6. 多帧数据处理结果汇总处理（思路一）

手势识别算法研究的手势样本为手势信息提取图，各图横轴均为时间轴，纵轴为各类具体信息，即距离、多普勒、角度等。

T 为帧周期，则时间轴向量为 t=(0:frame_num-1)*T。纵轴向量推算的具体公式此次不再赘述，直接给出结果。距离轴向量 R=c*(0:NSample-1)*Fs/2/freqSlope/NSample，速度轴向量 V=(-NChirp/2:NChirp/2 - 1)*lambda/Tc/NChirp/2，角度轴向量 A=asin((-Q/2:Q/2 - 1)*lambda/d/Q)*180/pi。

以上各参数的具体含义如图 18 所示。

```
frame_num=20;            %帧数
T=0.1;                   %帧周期
c=3.0e8;                 %光速
freqSlope=30e12;         %调频斜率
Tc=160e-6;               %chirp总周期
Fs=5e6;                  %采样率
f0=77e9;                 %初始频率
lambda=c/f0;             %雷达信号波长
d=lambda/2;              %天线阵列间距
Range_Number=256;        %采样点数/脉冲
NSample=Range_Number;    %距离向FFT点数
Doppler_Number=128;      %每帧脉冲数
NChirp=Doppler_Number;   %多普勒向FFT点数
Rx_Number=4;             %RX天线通道数
Tx_Number=1;             %TX天线通道数
Q = 180;                 %角度FFT点数
```

图 18

手势信息提取图为三维图，横轴和纵轴的含义根据以上讨论可知，其 Z 轴为幅值。因此绘制手势信息提取图的数据为一个二维矩阵，其大小为对应物理信息的 FFT 点数*帧数，则相应帧的数据为一个列向量。

多帧数据处理的过程较为简单，只需要将多个单帧处理返回的向量进行拼接即可。其处理代码如图 19 所示。

```
Range=zeros(NSample,frame_num);
Doppler=zeros(NChirp,frame_num);
Angle=zeros(Q,frame_num);
for xx=1:frame_num
    fdata = sdata((xx-1)*bufferSize+1:xx*bufferSize);
    [fft1d,fft2d,fft3d]=Three_dfft(fdata);
    Range(:,xx)=fft1d;
    Doppler(:,xx)=fft2d;
    Angle(:,xx)=fft3d;
end
```

图 19

处理结束后绘制手势信息提取图，其代码如图 20 所示。

```
t=(0:frame_num-1)*T;
figure();
R=c*(0:NSample-1)*Fs/2/freqSlope/NSample;
mesh(t,R,abs(Range));
xlim([0 max(t)]);ylim([0 max(R)]);
figure();
V=(-NChirp/2:NChirp/2 - 1)*lambda/Tc/NChirp/2;
mesh(t,V,abs(Doppler));
xlim([0 max(t)]);ylim([min(V) max(V)]);
figure();
A=asin((-Q/2:Q/2 - 1)*lambda/d/Q)*180/pi;
mesh(t,A,abs(Angle));
xlim([0 max(t)]);ylim([min(A) max(A)]);
```

图 20

7. 处理验证（思路一）

为了验证上述处理的正确性，假设只有一个目标，在单帧处理中添加 peakfind 函数从每帧数据中提取出其距离、速度和角度信息作为一个列向量 objres 输出，多帧数据汇总后绘制成二维图与手势信息提取图进行比较。

peakfind 函数代码分为两部分。第一部分为求出 angle_profile 的峰值对应的索引，第二部分是由索引根据相应物理信息的推算公式计算目标的距离、速度、角度。

计算峰值位置的代码如图 21 所示。

```
angle_profile=abs(angle_profile);
pink=max(angle_profile(:));
[row,col,pag]=ind2sub(size(angle_profile),find(angle_profile==pink));
```

图 21

计算目标的距离、速度、角度的代码如图 22 所示。

```
[Doppler_Number,Range_Number,Q]=size(fft3d);
c=3.0e8;                                                    %光速
fb = ((col-1)*Fs)/Range_Number;                             %差拍频率
fd = (row-Doppler_Number/2-1)/(Doppler_Number*Tc);          %多普勒频率
fw = (pag-Q/2-1)/Q;                                         %空间频率
d=lambda/2;                                                 %天线阵列间距
R = c*(fb-fd)/2/freqSlope;                                  %距离公式
v = lambda*fd/2;                                            %速度公式
theta = asin(fw*lambda/d);                                  %角度公式
angle = theta*180/pi;
objres=[R;v;angle];
```

图 22

将计算得到的目标的距离、速度、角度信息各自组合为一个大小为 1*frame_num 的向量，以向量 t 作为横轴，并对坐标设置与手势信息提取图相同的限制，绘制相应的二维曲线图。

某次数据绘制的二维曲线图和手势信息提取图如图 23~28 所示。

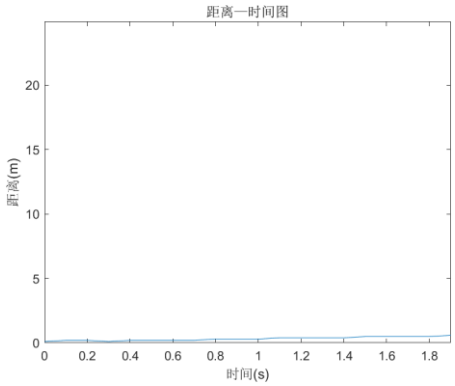

图 23

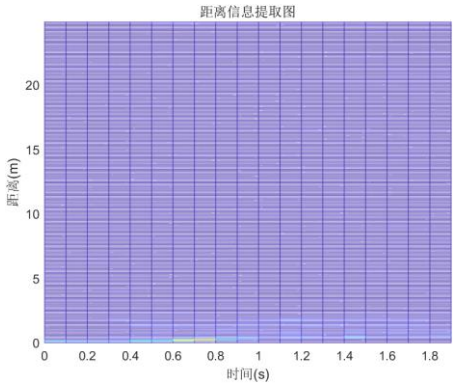

图 24

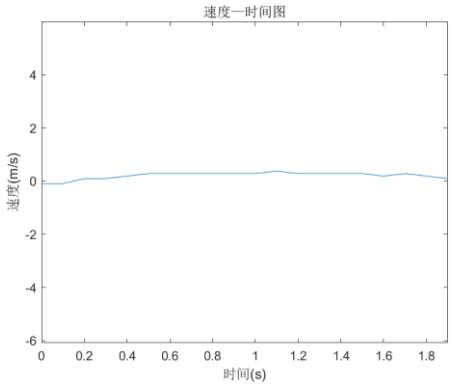

图 25

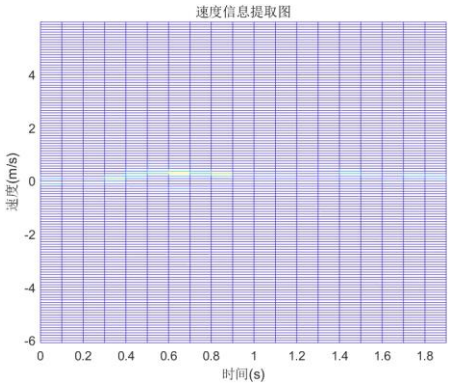

图 26

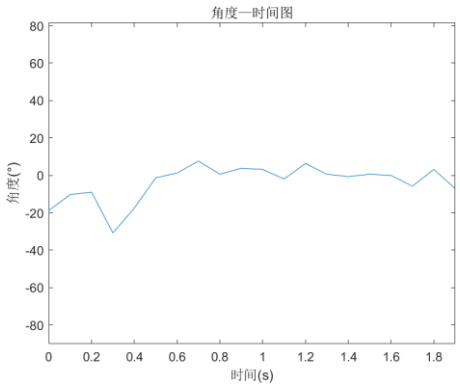

图 27

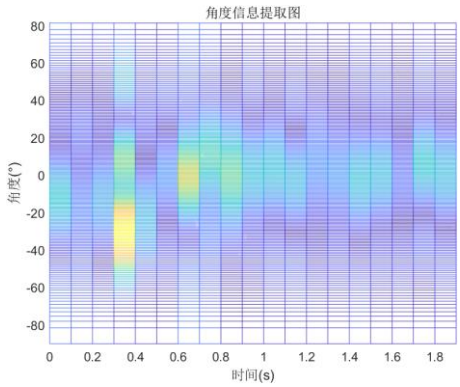

图 28

比较后可以发现，手势信息提取图的峰值和二维曲线图的纵坐标值的变化趋势和范围相当，多帧数据的处理方式在一定程度上是可信的。

8. 雷达配置（思路二）

StaticConfig、DataConfig 的配置与前文 2 部分相同，重点在于 SensorConfig 的配置。思路一以帧数代表时间轴，思路二以脉冲数代表时间轴。

profile 的配置如图 29 所示。其中修改脉冲采样率为 5MHz，采样点数为 512，而要满足这样的采样率和点数，必须增大脉冲持续时间，这里调整为 110us（顺带一提，《基于毫米波雷达的手势识别算法研究》一文中建议该值为 102.4us，恰等于 512/5M，但如此配置不被 mmWave Studio 所接收），脉冲空闲时间调整为 70us（不要过大即可），脉冲总时长为 180us。

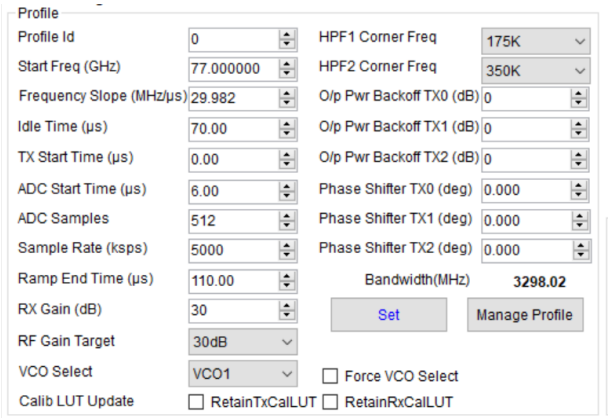

图 29

chirp 的配置如图 30 所示。使能天线 TX0 和 TX1，每根天线配置 200 个 chirp。

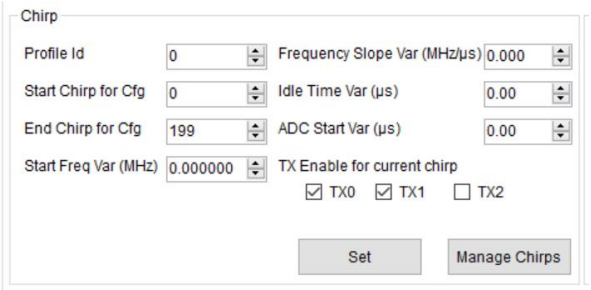

图 30

frame 的配置如图 31 所示。两根天线交替发射 chirp，每帧每 TX 各 200 个 chirp，共 20 帧，共计 8000 个 chirp。调整帧周期为 80ms（大于 200*2 个脉冲的总时长即可），则脉冲发射频率 PDF=200*2/80ms=5KHz，对应时间尺度为 8000/PDF=1.6s。

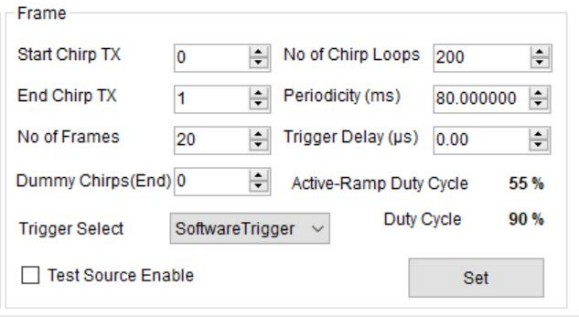

图 31

9. 数据处理流程（思路二）

这里我们将多个帧的原始数据一起进行处理，这样构成的三维数据矩阵与思路一明显的区别在于速度维的大小变大。

(1) 数据拆分、组合

第一步是将虚部和实部数据重组得到复信号。值得注意的是，LVDS Lane 1 只存放 I 路数据，LVDS Lane 2 只存放 Q 路数据，因此处理时每次重组 4 个数据点得到 2 个采样点的复信号。相关代码如图 32 所示。

```
fileSize = size(fdata, 1);
lvds_data = zeros(1, fileSize/2);
count = 1;
for i=1:4:fileSize-5
    lvds_data(1,count) = fdata(i) + 1i*fdata(i+2);
    lvds_data(1,count+1) = fdata(i+1)+1i*fdata(i+3);
    count = count + 2;
end
```

图 32

第二步是将复信号数据 lvds_data 排列成 3 维矩阵 ADC_Data(距离维、速度维、和天线维)，根据前文 3 部分对数据组织格式的讨论，容易得知 lvds_data 的数据格式如图 33 所示。

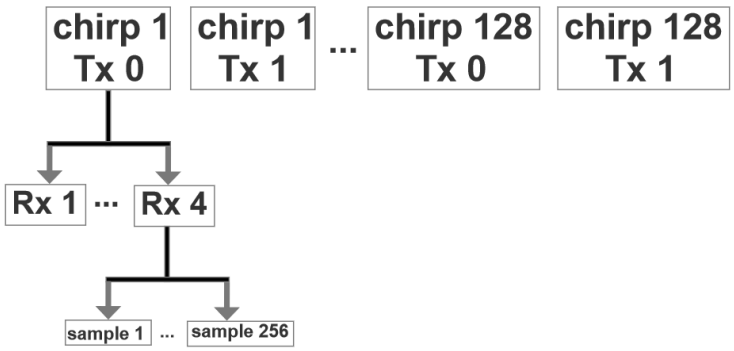

图 33

每个 chirp 有 Rx_Number 个通道，那么，每个通道的数据大小为 xSize=Range_Number，每个 chirp 的数据大小为 chirpSize=xsize*通道数。

根据以上讨论，可以得到排列数据的代码，如图 34 所示。

```
ADC_Data=zeros(Range_Number,Doppler_Number,Rx_Number); % 建立计算存储数据的空矩阵
% 数据组织格式:采样点(Sample)->天线(RX)->脉冲(Chirp)
xSize=Range_Number;
chirpSize=xSize*Rx_Number;
for i=1:Doppler_Number
    for j=1:Rx_Number
        ADC_Data(:,i,j) = lvds_data((i-1)*chirpSize+(j-1)*xSize+1:(i-1)*chirpSize+j*xSize);
    end
end
```

图 34

这些处理实际上与思路一并无太大区别，只是总 chirp 数（Doppler_Number）增多。此外，为了便于后文的绘图，我们将数据矩阵的第一维调整为距离维，第二维调整为速度维，这与思路一正好相反。

(2) 指数加权平均滤波

这里使用一种简单的指数加权平均滤波器进行杂波处理，其原理如图 35 所示。

$$\hat{s}(n) = s(n) - B_n$$

$$B_{n+1} = B_n + \alpha\hat{s}(n)$$

图 35

其中，$s(n)$为原始信号中第 n 个 chirp 回波，B_n为前 n-1 个 chirp 回波的加权积累，α为加权系数，$\hat{s}(n)$为经过滤波后的第 n 个 chirp 回波。

B 的积累等效于低通滤波，用于提取数据中的低频分量及 0 频分量，对应于场景中的缓动目标或静止目标，故通过从原始数据中将其减除，即可简便而高效地达到杂波抑制的目的。α越小，B 提取低频信号成分越大，则该方法抑制低频信号效果越强。

指数加权平均滤波的代码如图 36 所示

```
%% 指数加权平均滤波
B=0;      % B(n):前n-1个chirp回波的加权积累
a=0.01;   % 加权系数
% s(n)为原始信号中第n个chirp回波
% 滤波:s'(n)=s(n)-B(n);加权积累:B(n+1)=B(n)+a*s'(n)
for i=1:Doppler_Number
    ADC_Data(:,i,:)=ADC_Data(:,i,:)-B;
    B=B+a*ADC_Data(:,i,:);
end
```

图 36

(3) 距离信息提取

对 ADC_Data 的单个脉冲采样数据（距离维）进行 1D FFT 得到矩阵 range_profile，对各脉冲信息整合，得到整体的距离变化信息。这里加窗主要是为了使时域信号更好地满足 FFT 处理的周期性要求，减少信号的泄漏。

距离信息提取的代码如图 37 所示。

```
range_win = hamming(Range_Number);          % 加海明窗
range_profile = zeros(Range_Number,Doppler_Number,Rx_Number);
for j=1:Rx_Number
    for m=1:Doppler_Number
        temp=ADC_Data(:,m,j).*range_win;    % 加窗函数
        temp_fft=fft(temp,Range_Number);    % 对每个chirp做FFT
        range_profile(:,m,j)=temp_fft;
    end
end
```

图 37

(4) 距离维滤波

毫米波雷达的可探测距离较广，在手势识别应用场景中，雷达辐射范围比较广的话，可能会接收各种非目标手势回波，例如人体躯干的动作等等，而手势识别并不需要这些回波，可将其称为杂波，可以将这些杂波滤除。

可以利用如图 38 所示的距离维滤波器可以将人体动作、呼吸起伏滤除，只保留在手势范围内的回波。

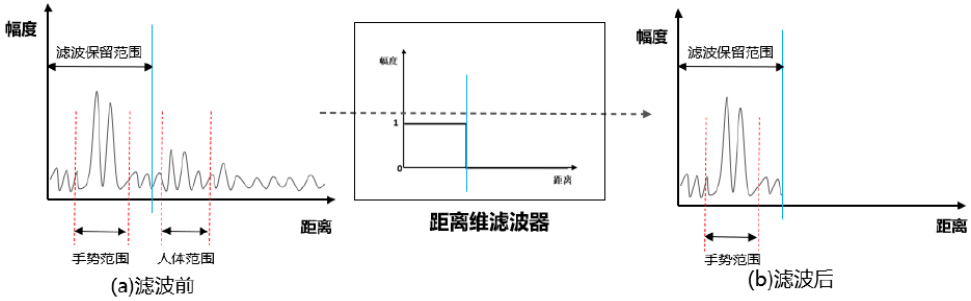

图 38

距离维滤波的代码如图 39 所示。

```
rlim=0.5;                                          % 手势范围
rlimid=ceil(rlim*2*freqSlope*Range_Number/c/Fs); % 距离为rlim对应的索引
range_profile(rlimid:end,:,:)=0;                   % 将距离>rlim的杂波滤除
```

图 39

(5) 多普勒信息提取

采用时频分析的方法来获取多普勒信息。时频分析中常用短时傅里叶变换，与普通的傅里叶变换相比，其更加注重瞬时频率信息。

STFT 的步骤是：首先，将信号与窗函数进行相乘，使得较长的回波信号在时间轴上分成几个固定时长的信号；然后，对每个时间窗内的信号进行一维 。通过不断滑动窗函数可以得到一系列傅里叶变换结果；最后，将结果排开即可得到整个信号的时变频谱。

将上述过程整合为功能函数 DopplerProcess，代码如图 40 所示。

```
function [speed_profile, F, T]=DopplerProcess(range_max,fs,win_sz)
window=hamming(win_sz);     % 使用海明窗
nfft = 2^nextpow2(win_sz); % DFT点数
nooverlap = win_sz - 4;     % 重叠单元长度
[speed_profile, F, T] = spectrogram(range_max, window, nooverlap, nfft, fs, 'yaxis');
end
```

图 40

其中采样率 fs 为脉冲发射频率，每次窗移动的步长为 4，不选用更小值的原因是减少计算量。

多普勒信息提取的代码如图 41 所示。

```
range_max=max(range_profile(:,:,1)); % 将距离信息中的各脉冲峰值位置数据提取出来，并重新排
win_sz=128;                          % STFT时间窗大小
[speed_profile, F, T]=DopplerProcess(range_max,PDF,win_sz);
F=fftshift(F);                       % 将零频分量移到频谱中心
F(F>=PDF/2)=F(F>=PDF/2)-PDF;         % 恢复对应频率
speed_profile=[speed_profile(win_sz/2+1:end,:);speed_profile(1:win_sz/2,:)]; % 将零频分量
```

图 41

先将距离信息 range_profile 中的各脉冲峰值位置数据提取出来，并重新排列成一列，然后对这列数据进行 STFT 操作实现时频分析处理从而获取手势的多普勒信息，具体实验设置为：将 STFT 的时间窗大小设定为 128 个数据点，重叠 124 个数据点。另外，返回的速度信息 speed_profile 和频率向量 F 应将零频分量移到频谱中心，恢复频谱（-PDF/2~PDF/2）。

(6) 角度信息提取

采用 MUSIC 算法提取目标的角度信息。

MUSIC 算法的基本步骤是：首先对不同接收天线接受到的雷达回波数据进行相关处理得到空间相关矩阵，然后对其空间相关矩阵进行特征分解得到一个信号子空间和噪声子空间。最后可以利用信号子空间与噪声子空间相互正交的特性构造一个伪谱函数，再进行谱峰搜索即可实现对目标角度的估计。

Music 算法的关键步骤的代码如图 42 所示。

```
function Pmusic=AngleProcess(X,lambda)
K=size(X,2);
Rxx=X*X'/K;                          % 计算协方差矩阵
[EV,D]=eig(Rxx);                       % 特征值分解
EVA=diag(D);                           % 将特征值矩阵对角线提取并转为一行
[~,I]=sort(EVA);                       % 将特征值排序 从小到大
EV=fliplr(EV(:,I));                    % 对应特征矢量排序 从大到小

% 遍历每个角度，计算空间谱
Pmusic = zeros(361,1);
derad = pi/180;
N = 4;                 % 阵元个数
M = 1;                 % 信源数目
dd=lambda/2;           % 天线间距
d=0:dd:(N-1)*dd;
En=EV(:,M+1:N);                        % 取矩阵的第M+1到N列组成噪声子空间
Hm=En*En';
for iang = 1:361
    angle=(iang-181)/2;
    phim=derad*angle;
    a=exp(-1i*2*pi*d/lambda*sin(phim)).';
    Pmusic(iang)=1/(a'*Hm*a);
end
end
```

图 42

其中，lambda 为雷达信号波长，X 则是由 8 个含有四通道信息的原始回波拼接而成（事实上可以每次只用一个回波信息，但这会增大计算量，而邻近回波的角度信息相差不大，因此每次取 8 个回波的信息是合理的）。

角度信息的提取代码如图 43 所示。

```
angle_profile=zeros(361,Doppler_Number);
for i=1:8:Doppler_Number
    X=reshape(ADC_Data(:,i:i+7,:),Range_Number*8,[]); % 将8个脉冲的数据拼接
    X=X.';                                             % 4(通道数)*4096(8*Range_Number)
    angle_profile(:,i:i+7)=AngleProcess(X,lambda)*ones(1,8);
end
```

图 43

(7) 绘制信息提取图

PDF 为脉冲发射频率，则时间轴向量为 t=(0:Doppler_Number-1)/PDF。距离轴向量 R=c*(0:Range_Number-1)*Fs/2/freqSlope/Range_Number；利用时频分析得到的频率 F，可以得到速度轴向量 V=F*lambda/2；实现 Music 算法的函数返回的角度信息列向量的长度是 361，对应-90°~90°，角度轴向量 A=linspace(-90,90,361)。

绘制信息提取图的代码如图 44、45 所示。

```
t=(0:Doppler_Number-1)/PDF;              % 时间坐标轴
figure();
R=c*(0:Range_Number-1)*Fs/2/freqSlope/Range_Number; % 距
mesh(t,R,abs(range_profile(:,:,1)));  % 绘制距离信息图
% xlabel('时间(s)');ylabel('距离(m)');
% title('距离信息提取图');
ylim([0 0.8]);                           % 限制坐标轴
view(2);                                 % 转换视线
set(gca,'position',[0 0 1 1]);           % 图像填满
figure();
V=F*lambda/2;                            % 速度坐标轴
mesh(T,V,abs(speed_profile));            % 绘制速度信息图
```

图 44

```
% xlabel('时间(s)');ylabel('速度(m/s)');
% title('速度信息提取图');
ylim([-1.2 1.2]);                        % 限制坐标轴
view(2);                                 % 转换视线
set(gca,'position',[0 0 1 1]);           % 图像填满
figure();
A=linspace(-90,90,361);                  % 角度坐标轴
mesh(t,A,abs(angle_profile));            % 绘制角度信息图
% xlabel('时间(s)');ylabel('角度(° )');
% title('角度信息提取图');
ylim([-50 50]);                          % 限制坐标轴
view(2);                                 % 转换视线
set(gca,'position',[0 0 1 1]);           % 图像填满
```

图 45

使用 view 来转换视线，忽略坐标轴信息，让图像填满 figure 的显示区域，便于后续对图像的操作。

10. 训练用手势信息提取图的获取

通过 9 部分的处理，我们已经获得了手势信息提取图，但要将其保存并用于后续的训练，还需要增添以下处理。

(1) 图像标准化

对图像样本进行标准化操作，具体而言包括尺寸标准化，及像素值标准化。

尺寸标准化，指数据集中图像尺寸大小应统一，依旧《基于毫米波雷达的手势识别算法研究》一文，将所有图像重新设置大小为 224*224。

像素值标准化，指对样本库中所有图像的像素值域进行统一规范。采用 minmax 标准化方式，将图片灰度值映射为[0, 1]范围内，具体而言，设某图片灰度值矩阵为 X，x_{ij}为单个像素点的灰度值，则$x_{ij} = \frac{x_{ij}-\min(X)}{\max(X)-\min(X)}$。

图像标准化的代码如图 46 所示。

```
function rimg=imstand(oimg)
 img=imresize(oimg,[224,224]);        % 尺寸标准化
 i=min(min(min(img)));
 j=max(max(max(img)));
 rimg=img-i/(j-i);                    % 像素标准化
end
```

图 46

(2) 图像和原始数据靠拷贝

由于每次获取的手势信息可能不够理想（做手势出现失误等），可以先将获取的图像使用 imwrite 函数（当然也可以直接写入到目标文件夹）保存在当前工作的文件夹下，待确认其可以作为训练使用后，将其拷贝到目标文件夹。

拷贝图像和原始数据的示例代码如图 47 所示。

```
function imcopy(fname,dstpath)
copyfile(fname,dstpath);              % 原始数据拷贝
copyfile('Range.png',dstpath);        % 距离信息图拷贝
copyfile('Doppler.png',dstpath);      % 速度信息图拷贝
copyfile('Angle.png',dstpath);        % 角度信息图拷贝
end
```

图 47

手势信息提取图的获取代码如图 48 所示。

```
oimg =getframe(1);
img=imstand(oimg.cdata);              % 图像标准化
imwrite(img,'range.png');
oimg = getframe(2);
img=imstand(oimg.cdata);              % 图像标准化
imwrite(img,'doppler.png');
oimg = getframe(3);
img=imstand(oimg.cdata);              % 图像标准化
imwrite(img,'angle.png');
```

图 48