Voice Localization Development Log: the 1st

Testing sensor quality and viability of planned method

Wenxuan Tang
12/5/2015

This Log is developing voice localization for squirtle.
Contains data collected during microphone testing.

Currently the solution is to use three microphones placed at three corners, using an arduino to send microphone strength(something indicating volume, need to find a value having highest robustness), to compute the angle of voice source.

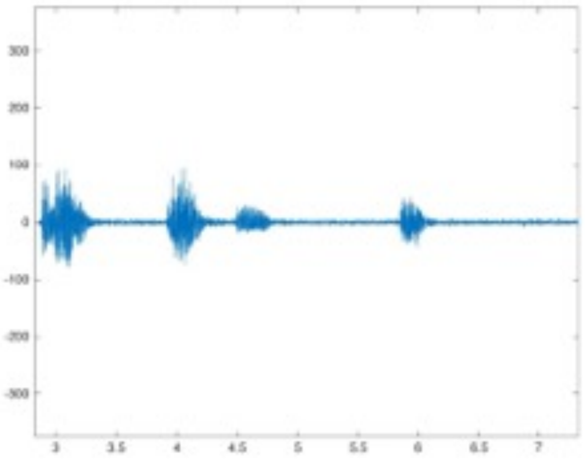
Uncertainty are:
We don't know what is the data collected by arduino like. We don't know whether the placement of microphone affect enough amount of microphone strength to allow us discriminate against noise.
We don't know what should the filter be in order to get a consistent and smooth "strength" value.

This log will aim at these questions.

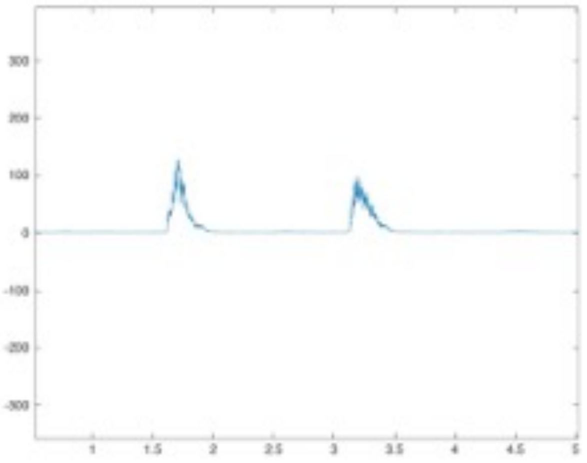
NOTE: the data shown here is not complete, instead we use descriptive language to describe the data. (yeah! we don't have that much time to write all of them down)

Single mic test:

Attach one microphone to arduino, collect raw microphone data and transmit them throw serial port, matlab receive them and plot out.



The upper image on the right is the raw data (note that we add a auto compute offset in code to make it around zero, originally its around 512), the data was collected by talking closely to mic.
We can see that the raw data noise level is acceptable(note that this has to be **<u>using 3.3V source</u>** on arduino, 5V is too noisy to use).

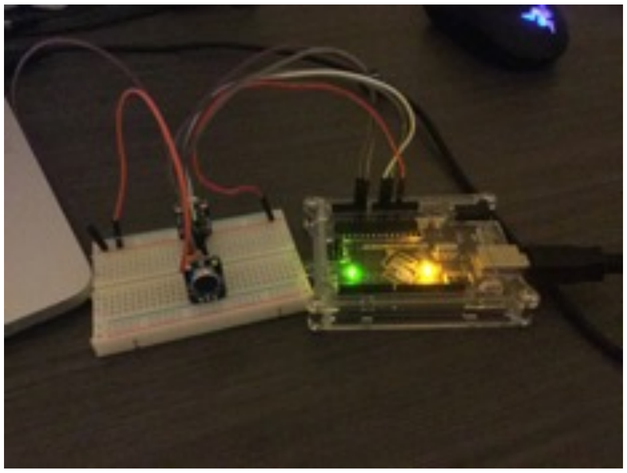


The lower image on the right is the filtered data. The filter is computing absolute value of raw data and then pass through a (0.1,0.9) low-pass filter.
We can see that filtered data is nice and clean (tho it's recorded in really close range, long range the peak decrease largely and thus signal-to-noise ratio decrease largely which is bad :( ). The delay of this 0.1 filter is acceptable since the data sampling frequency is very high.
Note that filtered data is always having a value larger than 0, which indicate that **<u>further calibration is needed.</u>**

Double mic test:

We put two microphone in opposite direction to test the difference of two microphones/ strength.

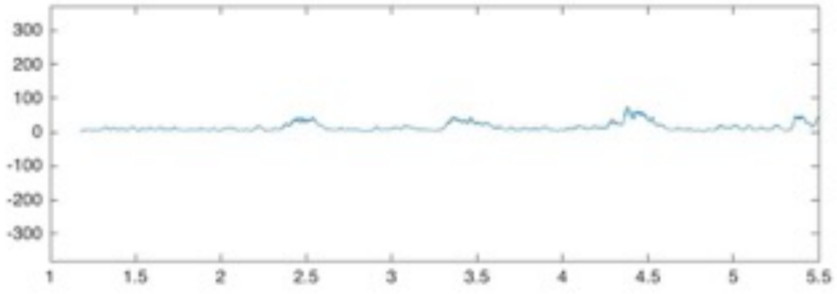
Sound Source was placed in 4 direction: facing mic1(expect mic1 having much higher strength), facing mic2(expect mic2 having much higher strength), facing middle of two sides(expect they equals).

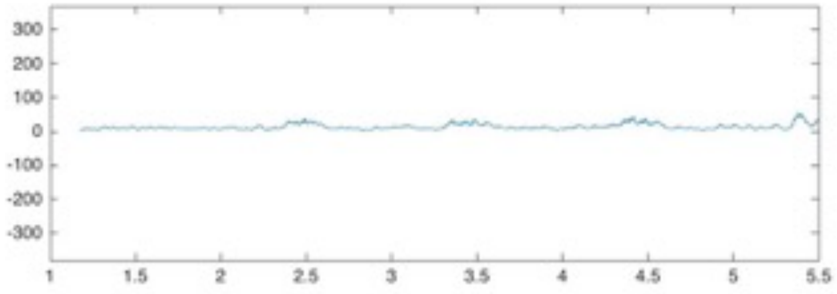


These two figures in collected by putting source facing mic1(upper) in middle range(around 1meter), and putting paper screen between the two mic (to increase their discrepancy).

We can see that the difference is barely (mic1 is a little larger).

Note that when without putting the screen the difference is not observable by eyes.(which indicate some hardware isolation is needed)





In fact , the two data has difference enough to discriminate with high robustness, if we handle it in a right way.
e.g.
in our test-facing-mic1 with close loud voice
the total value of mic 1 is 69577, and mic2 is 59355, which has high difference(around 20%)
in our test-facing-mic2 with close loud voice
the total value of mic 1 is 45466, and mic2 is 49355, which has high difference

Since we are only doing coarse test. The inconsistency of the two data is cause by voice source loudness, microphone gain, microphone offset, placing errors…. but the key idea here is that they DO HAVE difference when placed differently and that is observable (which is a good news for later development).

Also, we observe that mic1 is always having higher strength value than mic2 when we put the sound source in the middle(about 4% higher) . Which indicates that **further calibration needed** (this is probably cause by the gain of mic1 is larger)

Further, the later ROS voice localization is going to use strength of a specific time interval, we think that the **sum of all strength** is a better than **mean of all strength,** because the quiet strength will make two mic more undistinguishable, and really the whole amount of sound is better indicated in sum of them.

Conclusion:

From this test we know that:

Arduino can handle serial baud rate of up to 230400(maybe even higher), which is enough for continuity of sound data.

Current low-level low pass filter is not bad for our implementation, and this low-level filter can be executed in Arduino(speed is not bad),

Further calibration is essentially needed to :
    cancel offset (using quiet calibration)
    make scaling of each microphone equals (using directional calibration)
otherwise the mic array may failed to detect direction of sound source.

Use sum of strength in a specific time interval to metric sound source instead of mean.

Some hardware solutions needed to isolate microphones to have a good performance.

Potential problem exist in long range, especially with crowd of people.

Appendix A: Arduino transmitter code(single)

```
double raw_input = 0;
double offset = 512;
double static this_output = 0;
double static last_output = 0;

void setup() {
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
  Serial.begin(115200);
  double sum = 0;

  // compute the offset of data
  for (int i=0; i<10000; i++ ){
    sum += analogRead(A0);
    delay(0.1);
  }
  offset = sum/10000.0;

  // indicate the offset compute complete
  digitalWrite(13,HIGH);delay(500);
  digitalWrite(13,LOW);delay(500);
  digitalWrite(13,HIGH);delay(500);
  digitalWrite(13,LOW);delay(500);

}

void loop() {

  // compute the absolute value of raw data
  raw_input = analogRead(A0)-offset;
  if(raw_input<0)raw_input = -raw_input;

  // filter the raw data
  this_output = 0.1 * raw_input + 0.9 * last_output;
  Serial.println(this_output);
  last_output = this_output;

}
```

Appendix B: arduino transmitter code (double)

```
double raw_input_1 = 0;
double offset_1 = 512;
double static this_output_1 = 0;
double static last_output_1 = 0;

double raw_input_2 = 0;
double offset_2 = 512;
double static this_output_2 = 0;
double static last_output_2 = 0;


void setup() {
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
  Serial.begin(115200);
  double sum = 0;
  for (int i=0; i<10000; i++ ){
    sum += analogRead(A0);
    delay(0.1);
  }
  offset_1 = sum/10000.0;

  sum = 0;
  for (int i=0; i<10000; i++ ){
    sum += analogRead(A1);
    delay(0.1);
  }
  offset_2 = sum/10000.0;

  digitalWrite(13,HIGH);delay(500);
  digitalWrite(13,LOW);delay(500);
  digitalWrite(13,HIGH);delay(500);
  digitalWrite(13,LOW);delay(500);

}

void loop() {

  raw_input_1 = analogRead(A0)-offset_1;
  if(raw_input_1<0)raw_input_1 = -raw_input_1;

  this_output_1 = 0.1 * raw_input_1 + 0.9 * last_output_1;
  Serial.println(this_output_1);
  last_output_1 = this_output_1;

  raw_input_2 = analogRead(A1)-offset_2;
  if(raw_input_2<0)raw_input_2 = -raw_input_2;
  this_output_2 = 0.1 * raw_input_2 + 0.9 * last_output_2;
  Serial.println(this_output_2);
  last_output_2 = this_output_2;


}
```

Appendix C: matlab receiver code(single)

```
%% clean up
clear all;
newobjs = instrfind;
if ~isempty(newobjs)
    fclose(newobjs);
    delete(newobjs);
end
%% start up
s = serial('/dev/tty.usbmodem1451');
set(s,'BaudRate',115200);
fopen(s);
pause(5);

%% read data
out_datas = zeros(1,10000);
out_times = zeros(1,10000);
tic;
try
    for i = 1:10000
        out_line = fscanf(s,'%f');
        out_datas(i) = out_line;
        out_times(i) = toc;
    end
catch exception
    warning('read failed');
    msgString = getReport(exception);
    warning(msgString);
end
%% cleanup
fclose(s);
delete(s);

%% plotting
figure(1);
plot(out_times,out_datas);
axis([1,10,-500,1000]);
```

Appendix D: matlab receiver code(double)

```
%% clean up
clear all;
newobjs = instrfind;
if ~isempty(newobjs)
    fclose(newobjs);
    delete(newobjs);
end
%% start up
s = serial('/dev/tty.usbmodem1451');
set(s,'BaudRate',115200);
fopen(s);
pause(5);

%% read data
out_datas_1 = zeros(1,10000);
out_times_1 = zeros(1,10000);
out_datas_2 = zeros(1,10000);
out_times_2 = zeros(1,10000);


tic;
try
    for i = 1:10000
        out_line = fscanf(s,'%f');
        out_datas_1(i) = out_line;
        out_times_1(i) = toc;
        out_line = fscanf(s,'%f');
        out_datas_2(i) = out_line;
        out_times_2(i) = toc;

    end
catch exception
    warning('read failed');
    msgString = getReport(exception);
    warning(msgString);
end
%% cleanup
fclose(s);
delete(s);

%% plotting
figure(1);
subplot(2,1,1);
plot(out_times_1,out_datas_1);
axis([1,10,-500,1000]);
```

```
subplot(2,1,2);
plot(out_times_2,out_datas_2);
axis([1,10,-500,1000]);

(sum(out_datas_1) - sum(out_datas_2))/sum(out_datas_2)
```