Pers Ubiquit Comput (2014) 18:1753–1762
DOI 10.1007/s00779-013-0706-7

ORIGINAL ARTICLE

# Classifying watermelon ripeness by analysing acoustic signals using mobile devices

**Wei Zeng · Xianfeng Huang · Stefan Müller Arisona · Ian Vince McLoughlin**

Received: 28 August 2012 / Accepted: 6 July 2013 / Published online: 6 August 2013


**Abstract** This work addresses the problem of distinguishing between ripe and unripe watermelons using mobile devices. Through analysing ripeness-related features extracted by thumping watermelons, collecting acoustic signals by microphones on mobile devices, our method can automatically identify the ripeness of watermelons. This is possible in real time, making use of machine learning techniques to provide good accuracy. We firstly collect a training dataset comprising acoustic signals generated by thumping both ripe and unripe watermelons. Audio signal analysis on this helps identify features related to watermelon ripeness. These features are then used to construct a classification model for future signals. Based on this, we developed a crowdsourcing application for Android which allows users to identify watermelon ripeness in real time while submitting their results to us allowing continuous improvement of the classification model. Experimental results show that our method is currently able to correctly classify ripe and unripe watermelons with an overall accuracy exceeding 89 %.

**Keywords** Mobile and physical computing · Real-time signal analysis · Machine learning · Crowdsourcing

W. Zeng (✉) · X. Huang · S. Müller Arisona
Future Cities Laboratory, Department of Architecture, ETH Zurich, 8093 Zurich, Switzerland
e-mail: zeng@arch.ethz.ch

I. V. McLoughlin
University of Science and Technology of China, Hefei, China

## 1 Introduction

The rich set of sensors such as microphone, camera, GPS and accelerometer on mobile devices enables them to interact with their environment in many ways. Applications making use of sensors on mobile devices have introduced a new area of research called mobile sensing. The potential of mobile sensing has been discussed for a long period in both industrial and research communities. For instance, Nokia has developed the Nokia Remote Sensing platform (NORS) which aggregates mobile phone data and notifies changed sensor data for a variety of application scenarios such as health care and environmental monitoring [4]. Such efforts have led mobile devices to play an increasingly important role in the daily lives of the public, relating to communication, social activities, healthcare and so on.

Among mobile device sensors, the microphone is of course the most ubiquitous but perhaps least exploited for non-vocal applications. Sounds collected by a mobile device's microphone can be analysed to make accurate inference regarding the carrier, the environment and even the current social setting [10]. Apart from just recording sounds, the microphone included in every mobile device can be used for a variety of purposes [14]. The development of SDKs and libraries supporting acoustic signal analysis techniques on recent mobile operating systems (such as iOS, Android and Windows Phone) brings more possibilities for applications making use of microphones on mobile devices.

For acoustical signal analysis, the key problem is normally to find features that can be used for recognition and classification. Past researchers have extracted many features from acoustic signals that are reviewed in [15]. The features are summarised and classified into seven domains:

temporal, physical frequency, perceptual frequency, cepstral, modulation, eigendomain and phase space, with each domain consisting of several specific features. Once features are extracted, some form of recognition or classification method is usually required for the application of interest. Automatic acoustic content recognition and classification methods are summarised in [16]. Typical methods include Bayesian classification, nearest neighbour classification, hierarchical clustering, hidden Markov model (HMM) and support vector machine (SVM).

Considering the particular application domain of this paper, experienced farmers and buyers normally thump watermelons to judge their ripeness before picking or buying them. When thumping, they listen to the sound produced and claim the ability to distinguish between ripe and unripe fruits. This suggests that there exist acoustic features related to the ripeness of watermelons, and that we may be able to use those features to automatically judge whether watermelons are ripe or not.

In this paper, we discuss the implementation of a crowdsourced application that allows us to collect watermelon acoustic data while simultaneously providing a tool to farmers and buyers that will assist them in the automatic detection of ripe fruit. The benefit of making it crowdsourcing is that we can collect watermelon acoustic data from different mobile devices, users and watermelons from many countries allowing continuous improvement of our classification model.

In our work, we firstly collect a training dataset containing acoustic signals generated by thumping both ripe and unripe watermelons. Experiments are done to retrieve features related to the ripeness. A real-time crowdsourcing mobile application able to judge watermelon ripeness automatically using mobile devices is implemented, and its performance is evaluated.

Contributions of our work are:

- We find a series of key acoustic features related to ripeness of watermelon, using relatively low-complexity measures including zero crossing rate, short-time energy and other.
- We show how mobile devices can easily implement the analysis in real time, and how this can be made accessible to users and to implementers through crowdsourcing for improving the classification method.
- Our experiments show that our proposed method is able to correctly classify ripe and unripe watermelons with an overall accuracy of 89.9 %.

The remainder of this paper is organised as follows. We show related work in Sect. 2. System, mathematical model and user interaction of our method are discussed in Sect. 3. Section 4 explains details of our methods followed by implementation of a mobile application in Sect. 5. Experiments results are listed and evaluated in Sect. 6. Finally, we conclude and discuss future work in Sect. 7.

## 2 Related work

The watermelon is an important and common fruit, especially during summer and in tropical areas. Purchasers are concerned most with their quality and ripeness, and thus, experienced buyers will often thump several watermelons to judge their ripeness according to some kind of heard or felt acoustic response. However, experience is needed for this, and it is difficult for people with less-gifted ears or in noisy market environments. Thus, there is a need for an easy implementable, low cost and portable method that can automatically judge ripeness. In fact, some related research has been published previously regarding the quality and ripeness of watermelons.

### 2.1 Watermelon quality

Several internal defects can cause bad watermelon quality, including (a) creases or large voids, (b) over-ripeness and (c) bruises, usually due to impacts [7]. Based on the hypothesis that acoustic impact response of bad quality watermelons differs to those good quality, Diezma-Iglesias et al. [7] constructed an apparatus consisting of a microphone, structural elements and a mechanical impact generator to non-destructively detect defects in watermelons based on acoustic impulse response. In this work, spectral parameters are obtained by summing the magnitude of the spectrum between two frequencies in a specified band width. This is repeated for good and defective seedless watermelons, and the acoustic parameters are then used for classification purposes.

### 2.2 Watermelon ripeness

Watermelons do not continue to ripen after harvest, and hence, it is critically important to judge whether they are ripe or not prior to picking. Although 90 % of the watermelon is water, sugar is an important indicator in the process of ripening. Sugar content will increase when watermelons become ripe, and this will lead to sound frequencies that differ in acoustic response from unripe watermelons [6].

Therefore, various studies have been undertaken on acoustic impulse response relationship to watermelons ripeness [2, 3, 18]. In fact, Baki et al. [3] analysed Mel-frequency cepstrum coefficients (MFCC) derived from acoustic signals collected from ripe and unripe watermelons. The coefficients are used to train a multi-layer perceptron (MLP), and a best MLP classifier is constructed

**Fig. 1** Internal view comparison between ripe (*left*) and unripe (*right*) watermelons

from MLP structures and parameters to classify watermelon ripeness. Baki et al. [3] reported that their method is able to discriminate between ripe and unripe watermelon with an accuracy of 77.25 %.

However, much of this research requires expensive and complicated mechanical devices and experimental apparatus, which tends to make their methods less useful for normal users, such as those shopping in a supermarket. Also, the accuracy of their results is not particularly high. In contrast to this prior work, we analyse the acoustic signals in more detail and define more ripeness-related features. It should also be noted that although there are some applications, such as iWaterMelon, MelonMeter, WaterMelon Prober, which claim to help users classify watermelons ripeness with mobile devices, we found these to be unreliable as well as unpublished. These applications are neither sensitive to the acoustic response of thumping watermelons nor accurate enough.

Hence, we develop a technique which makes use of current widely available mobile devices to record acoustic signals generated by thumping watermelons and classify their ripeness in real time. Our focus is on distinguishing ripe watermelons from unripe ones as depicted in Fig. 1. And we are mostly concerned with choosing and purchasing watermelons. Other factors relating to watermelon quality are not covered in our study. Currently, we only implement our method for Android, but it is easy scalable to all mobile platforms.

## 3 Overview

### 3.1 System overview

We use machine learning techniques to classify watermelon ripeness automatically. Our method contains two phases—a training and a classification phase—as illustrated in Fig. 2. In both phases, recorded acoustic signals are firstly processed to separate background noise and segment thumping frames. Then, ripeness-related features are extracted from each thumping frame. For training data, the features are labelled according to the subsequently known ripeness of the watermelons in the training phase. These are used to train a classification model which is applied during the classification phase to identify ripeness of other watermelon signals.

Users can contribute to the implementation by sending us the feedback of classification result. Combined with raw audio data, our system is able to automatically update the training model. With the crowdsourcing capability, our implementation overcomes the limitation of relatively small amount of experiment data and limited range of watermelon types.

### 3.2 Mathematical model

Factors of watermelon quality other than ripeness, like creases and voids, can also affect the thumping acoustic

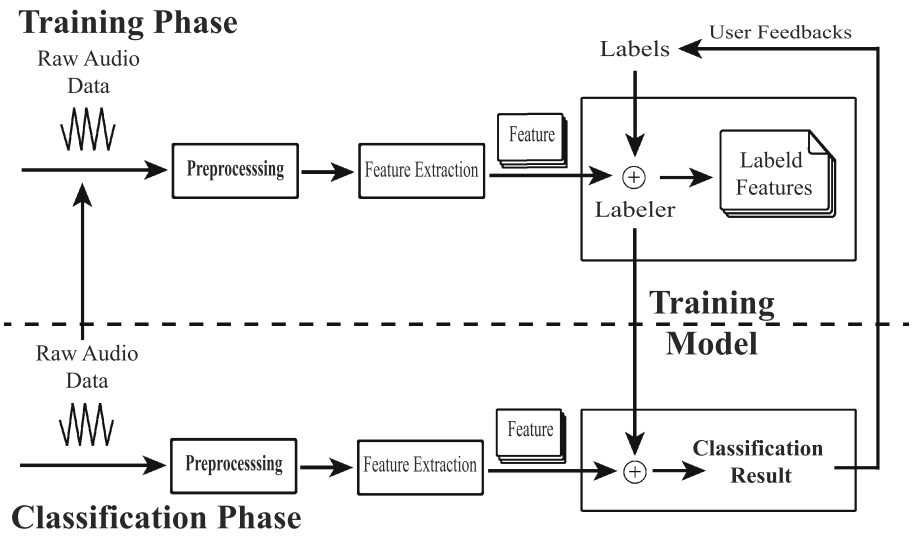


**Fig. 2** System overview of our method

signals. The acoustic signals $W_{\text{melon}}$ collected can be formulated as follows

$$W_{\text{melon}} = f(W_{\text{state}}, D_{\text{acc}}, \text{Force}) + \text{Noise} \quad (1)$$

where $W_{\text{state}}$ is the state of watermelon, including size, shapes, quality and others. $D_{\text{acc}}$ is the gain and frequency response of the microphone on the mobile devices, which varies with different mobile devices models and usage periods. Force represents the force users applied to thump the watermelons (by hand), and noise represents the environmental and system noise.

In general, ripe watermelons usually sound dull when thumped while unripe ones have a tighter, metallic sound. This is regardless of other factors, such as how much thumping force is used. We know that the microphones on mobile devices offer signal capture of considerable quality [14] and therefore expect that they are sufficiently accurate to distinguish between the two types of signal. Based on the hypothesis that acoustic response signals of thumping watermelons are mainly affected by ripeness, we built a classification model based on watermelon ripeness-related features.

### 3.3 User Interaction

After generating the classification model, we implemented a mobile application that is able to classify watermelon ripeness in real time. Figure 3 illustrates how users utilise our application on mobile devices. Users should hold their mobile devices running our application near to the watermelon surface in one hand while thumping it with another hand. The acoustic signal generated by thumping will be reflected internally and collected by the microphone. Following detection of a single thumping event, our application will analyse the signals and indicate watermelon ripeness in real time.

## 4 Methods

Our method is basically a machine learning technique and consists of three parts: preprocessing, feature extraction and classification. Preprocessing is designed to separate the thumping response from background noise and segment thumping event frames. Watermelon ripeness-related features are extracted in the feature extraction step, and finally, these features are classified.

### 4.1 Preprocessing

Collected acoustic signals are preprocessed to separate noises and segment thumping event frames. Each frame indicates one individual watermelon thumping event and contains acoustic watermelon ripeness-related features. Basic hypothesis is that signal-to-noise ratio (SNR) is high enough to distinguish thumping signals from noise. In practice, this is not difficult to achieve. Figure 4 illustrates steps of preprocessing raw acoustic response signals.

#### *4.1.1 Calculate root mean square (RMS)*

RMS is an effective method to separate noise from event signal, especially when SNR is high. RMS is calculated by taking frame of the acoustic signal and computing the square root of the sum of the squares of the windowed sample values. It can be formulated as

$$\text{rms} = \sqrt{\frac{1}{n}\sum_{i}^{n} s_i^2} \quad (2)$$

where $n$ is window size and $s(i)$ is the input acoustic signals.

Window size is one of the key factors affecting the signal-noise separation result. It needs to be short enough to accurately locate the start and end points of the event signal yet long enough to contain a recognisable event signal. Overlapping frames are proposed in [12] to more precisely capture subtle changes in the acoustic signals (as is common in speech processing); however, this will also increase the computing overhead. In order to accelerate computing speed and reduce power consumption for

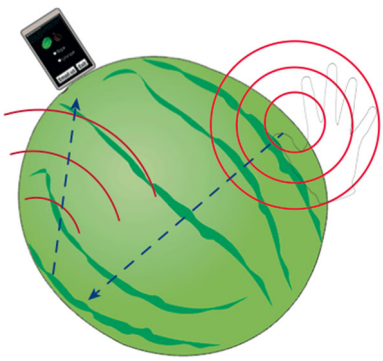

**Fig. 3** Thump a watermelon and record acoustic signals using the microphone on the mobile device

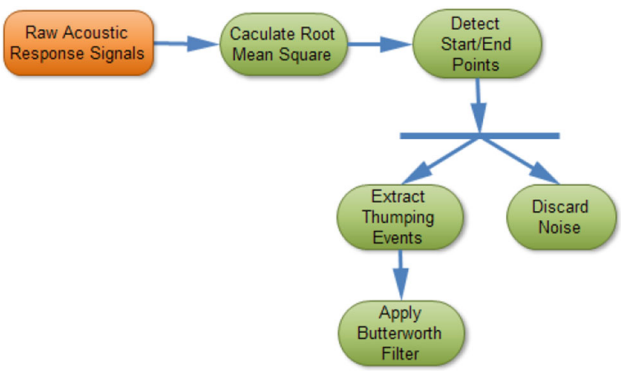


**Fig. 4** Work flow of preprocessing of acoustic signal

mobile devices use, we adopt a fixed frame size of 1 ms (44 samples). We observed that each watermelon thumping signal lasts about 40 ms.

#### 4.1.2 Detect start/end points

Frames are characterised into 0 or 1 based on their RMS energy. Frames with RMS values below threshold are labelled 0, and those above are labelled 1. Transitions from 0 to 1 are considered start points, and transitions from 1 to 0 are considered end points.

It is difficult to determine a fixed threshold value since noise levels vary in different situations and with different microphones. Hence, we use a dynamic threshold: RMS values in a short period (2 s) before the users thump the watermelon are determined, and five times this value is used as threshold.

#### 4.1.3 Extract thumping event

Signals between consecutive start and end points are considered to contain a potential thumping response. We observed that all watermelon thumping signal lengths are about 1,800 samples (41 ms). Hence, we only take into consideration frames with size between 1,500 and 2,500 samples (34–57 ms). For short but close frames (each frame size less than 1,000 and distance less than 500), we merge them together and consider them as originating from the same event. After filtering, all frames satisfying requirements are considered thumping events.

#### 4.1.4 Discard noise

All other signals are considered noise and discarded.

#### 4.1.5 Apply Butterworth filter

In order to further reduce unwanted noise effects, we applied a second order low pass Butterworth filter on extracted signals to reject unwanted frequencies [5]. We found that environmental noise mostly lay in the higher part of frequency domain, and hence, we define a cut-off frequency at half of the Nyquist frequency.

After all these steps, acoustic signals containing watermelon thumping events have been separated from much of the noises and segmented into frames. Figure 5 shows the preprocessing results of an acoustic response signal generated by thumping on a ripe watermelon.

### 4.2 Feature extraction

After noise reduction and segmentation, a classification model is constructed based on watermelon ripeness-related features. Feature selection is critical to this step, and thus, we analysed the following features as suggested in [13] in order to find watermelon ripeness-related features.

#### 4.2.1 Zero crossing rate (ZCR)

ZCR is defined in [17] as the number of time-domain zero-crossings within a frame. It is commonly used in audio signal processing, especially for classifying percussive sounds [8]. ZCR can be calculated as

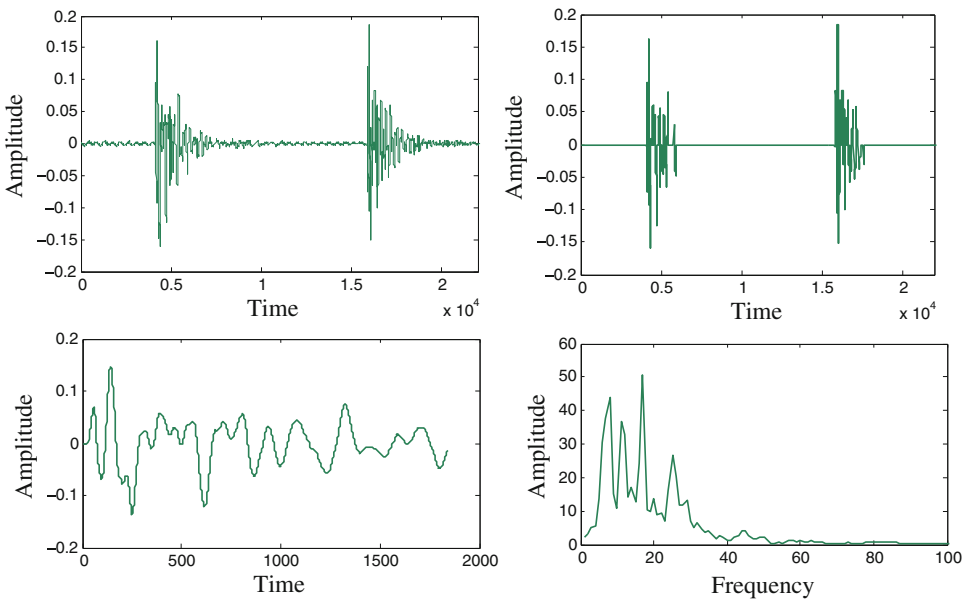


**Fig. 5** Preprocessing of raw acoustic impulse response from thumping a ripe watermelon: raw acoustic signal (*top left*); noise removed acoustic signal (*top right*); frame of patting event extracted (*bottom left*); fast Fourier transform of the frame (*bottom right*)

$$\text{zcr} = \frac{1}{2N}\sum_{i=1}^{N}|\text{sign}(s_i) - \text{sign}(s_{i-1})| \quad (3)$$

where $N$ indicates frame length, *sign*() function is 1 for positive inputs and $-1$ for negative inputs, $s_i$ represents input signals.

#### 4.2.2 Short-time energy (STE)

STE measures the sum of the square of signals in time domain that can be formulated as:

$$\text{ste} = \sum_{i=1}^{N} s_i^2 \quad (4)$$

where $N$ indicates frame length and $s_i$ represents input signals. Similar to ZCR, STE is simple to compute, but it is an important feature in audio signals.

#### 4.2.3 Sub-band short-time energy ratio

Sub-band STE ratio measures the ratio between the sub-band energy within intervals and the total energy in a frame. It is formulated as:

$$\text{sub\_ste\_ratio} = \frac{1}{\text{ste}}\sum_{i=L}^{H} s_i^2 \quad (5)$$

where $L$ and $H$ are the lower and upper sub-band boundaries, $s_i$ is the input signal. Sub-bands are normally designed according to real situations. In our case, we divide frames into sub-bands of $\left[0, \frac{T}{8}\right]$, $\left[\frac{T}{8}, \frac{T}{4}\right]$, $\left[\frac{T}{4}, \frac{T}{2}\right]$ and $\left[\frac{T}{2}, T\right]$. We observed that unripe watermelon STE mostly falls in the first sub-band. Also that the sub-band STE ratios drop fast while for ripe watermelons, the sub-band STE ratios are much flatter as depicted in Fig. 6.

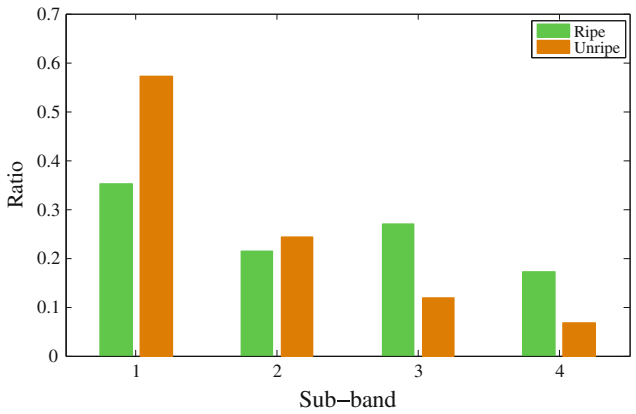


**Fig. 6** Sub-band STE ratio of typical ripe and unripe watermelons acoustic signals

#### 4.2.4 Mel-frequency cepstral coefficients (MFCC)

MFCC are commonly used features in audio signal processing, especially for speaker recognition. For selected frames, MFC coefficients constitute good representations of dominant features. They are normally calculated as following:

- Select a frame of samples from the original signal.
- Apply Fourier transform to the obtained signal frame.
- Apply Mel-frequency filtering to the transformed signal.
- Take logarithm of the powers at each Mel frequency.
- Take discrete cosine transform (DCT) of the list of Mel log powers.

This can be formulated as:

$$c(n) = \text{DCT}(log(|\text{FFT}(s_i)|) \quad (6)$$

where $c(n)$ is the MFC coefficients and $s_i$ is the input signal. FFT and DCT indicate fast Fourier transform and discrete cosine transform respectively.

#### 4.2.5 Brightness

Brightness is defined as the centroid of the frequency spectrum in a frame which can be computed as:

$$w_c = \frac{\sum_{w=0}^{B} w|F(w)|^2}{\sum_{w=0}^{B}|F(w)|^2} \quad (7)$$

where $B$ is the Nyquist frequency, $F(w)$ is the value of FFT at frequency component $w$.

#### 4.2.6 Spectrogram

A spectrogram reveals how spectral density varies with time. The spectrogram of a signal is normally estimated by computing a sequence of short-time Fourier transforms (STFT) of the signal, with squared magnitude. This can be formulated as:

$$\text{Spectrogram}(t, w) = |\text{STFT}(t, w)|^2 \quad (8)$$

where *STFT* indicates short-time Fourier transform at time $t$ and frequency $w$.

Figure 7 shows a comparison of the time-frequency representation of ripe and unripe watermelon thumping response signals. Note that only part of the frequency domain is shown as values in the remaining portion are very low. However, the figure shows clearly that density is higher in both the low part of time and frequency domain of unripe watermelon thumping response signals, which indicates that their *brightness* is lower and their *sub-band STE ratio* drops faster.

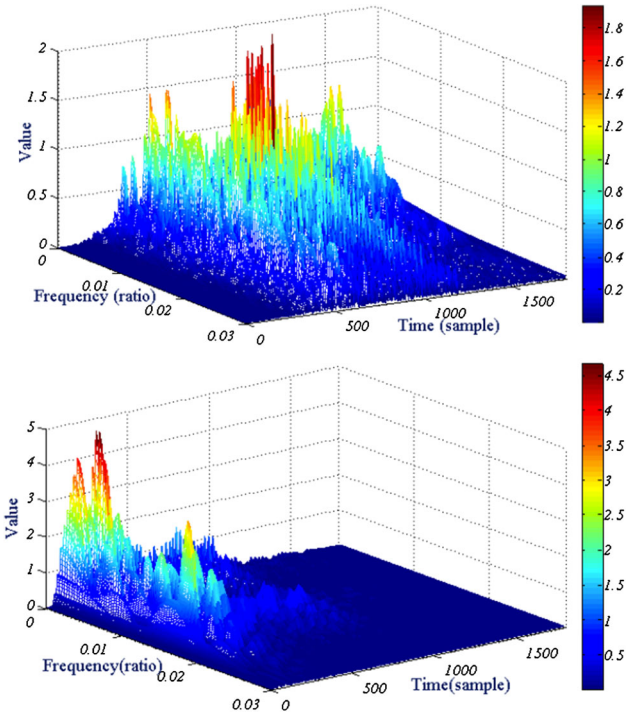


**Fig. 7** Comparison of time frequency representation of ripe (*top*) and unripe (*bottom*) watermelons acoustic response signals

### 4.3 Classification method

Many methods have been proposed to classify audio streams, including K-nearest neighbour (KNN), Gaussian mixture model (GMM) and support vector machine (SVM). We explored the performance of these methods in audio segmentation and classification and chose SVM. It is claimed in [11] that SVM-based methods can outperform both KNN and GMM for similar tasks.

Given a training dataset that is marked as positive or negative, the SVM training algorithm learns an optimal separating hyper plane, which assigns a new example as positive or negative [1]. The optimal separating hyper plane is achieved by making the largest distance to the nearest training data points of both classes.

SVM can be either linear or nonlinear (kernel based), where the difference is that nonlinear SVM needs to map the data into a high dimensional feature space to make the data linearly separable. Normally, nonlinear SVM has a better performance than linear SVM, but it also requires more computing power. Taking into consideration, the limitations of mobile devices and to balance between the performance and computing time, we constructed a linear SVM in our implementation.

Training data are represented as $(x_i, y_i)$ where $x_i \in R^n$ is a feature vector consisting of $n$ watermelon ripeness-related features and $y_i \in \{-1, +1\}$ is a class label. Each feature vector is labelled either $-1$ or $+1$, representing unripe and ripe, respectively. Therefore, we constructed a classification model with a hyper plane separating ripe from unripe feature vectors as much as possible. This classification model is further tested by classifying testing the dataset with result presented in Sect. 6.

## 5 Implementation

### 5.1 Implementation on Android devices

We implemented our method on Android due to its very wide spread adoption and operation on multiple platforms. However, Android is Java based and not particularly well suited to real-time or low-latency applications, especially for audio processing. In order to achieve real-time and low-latency processing, we followed the framework proposed in [9]. Juillerat et al. [9] stated that the following things should be taken into account for Java audio processing: the audio pipeline, audio API, HotSpot compiler, garbage collection and priority inversion. Hence, such techniques are used in our implementation:

- We create one thread for audio recording by which audio is recorded in a mono channel, with 16 bits per sample at 44.1 kHz sampling frequency. A ring buffer is allocated for temporally storing audio signals with minimum buffer size (8,192 bytes for our test phones) calculated by the Android system.
- We create one thread for processing the recorded audio signal. Our application first collects a period (2 s) of environmental sounds to calculate an RMS threshold for separating noise from thumping event. Signals are considered as environmental noise and discarded if their RMS is below the RMS threshold. A Butterworth filter is applied to signals with RMS above the threshold since they are considered to contain thumping events. Feature vectors are extracted from these signals and labelled as ripe or unripe using the classification model we constructed. A final classifying decision is made by a majority vote from these labels.

With such steps, we achieved a mobile application able to judge watermelon ripeness in real time with relatively low latency. However, the actual performance also depends upon environmental noise, microphones quality, mobile device computing speed, memory and background tasks. We also created one additional thread for users to submit their collected signals if they find the result is wrong and wish to provide this information. This will cause an email to be sent to us containing collected audio signals and correct result. This crowdsourcing ability helps us collect more data points and improve our classification model continuously.

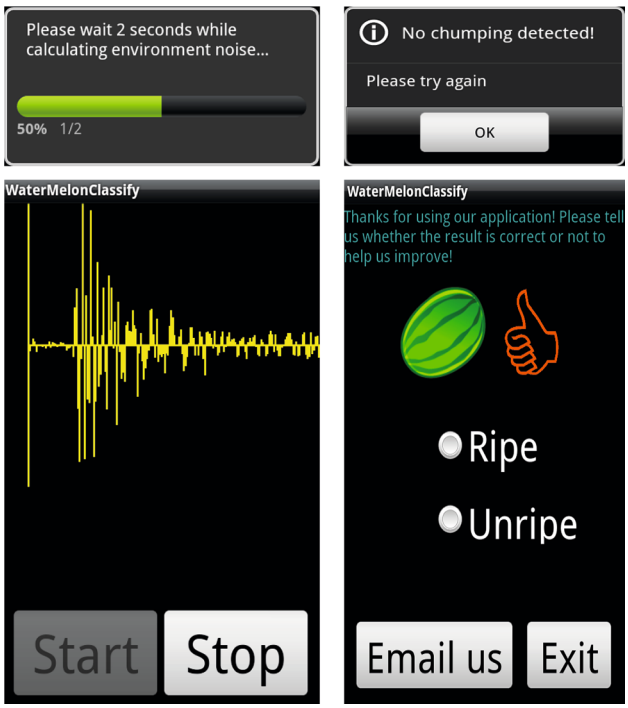


**Fig. 8** User interface of our application. *Top left* dialogue asking users to wait 2 s for measuring environment noise; *bottom left* screen showing the acoustic signal collected after the user press *Start* button; *top right* alert informing users no thumping signals are detected; *bottom right* result screen showing that the tested watermelon is ripe and asking users permission to send us the result for improving our classification model

### 5.2 User interface

We developed a user interface that is simple and easy to use as shown Fig. 8. A dialogue (top left) asking users to wait for two seconds in order to measure environment noise will be shown to users when the application starts. Then, users start collecting thumping signals by pressing the *Start* button, and signals will be presented to the users (bottom left) in near real time. The background thread will start recording and analysing at the same time. Users press the *Stop* button after they have thumped their watermelons. If our application cannot detect any thumping signals, a warning will be shown (top right). Otherwise the classification result (bottom right) will pop up. A thumbs up figure indicates the watermelon is ripe while thumbs down means the watermelon is unripe. Users may give us feedback to improve the classification model at this point, or they can simply *Exit* the application.

## 6 Experiments

The training dataset is generated by patting on 10 ripe and unripe watermelons, respectively. Thumping acoustic response signals are all recorded as discussed in Sect. 4 and used to construct a classification model making use of SVM algorithms with a hyper plane separating the two classes most widely. We then test the classification model on a testing dataset with 40 watermelons, of which 15 are ripe and 25 are unripe. Tests used several models of mobile phone running Android version 2.1 and above, and were in two locations: a farming field and a quieter laboratory environment.

### 6.1 Thumping detection

Table 1 provides the number of ripe and unripe watermelons used in training and testing dataset, respectively, with the total number of thumping events detected (i.e. multiple thumps per watermelon) and ratio of correct detections in the brackets. We can see from the figure that our technique is able to detect most thumping acoustic response signals from noise. We can also see that the performance is better for ripe watermelons because we bought them and collected the acoustic signals in a quite laboratory environment. By contrast, we did not purchase the unripe watermelons, and thus, all thumping tests were recorded in the field with higher environment noise. Nevertheless, the detection performances are acceptable in both cases.

### 6.2 Features analysis

Both time and frequency domain features are extracted from frames containing thumping events. However, taking into consideration of limited computing power of mobile devices in relation to the complexities of *MFCC* and *spectrogram*, these two features were not adopted for the final feature vector. However, there is research [3] showing that *MFCC* can generate results with an accuracy of around 77 %.

Table 2 shows the mean values of features we adopt in the model of both ripe or unripe watermelons from the training and testing datasets. From these values, some conclusions can be drawn:

1. Ripe watermelons have lower values of *ZCR* than those of unripe watermelons.
2. Ripe watermelons have lower values of *Short-Time Energy* than those of unripe watermelons.

**Table 1** Training and testing datasets: number of watermelons, number of thumping events detected and correct detections ratio

| | Training set | Testing set |
|---|---|---|
| Ripe | 10 / 56 / 98.2 % | 15 / 114 / 97.4 % |
| Unripe | 10 / 42 / 93.3 % | 25 / 160 / 91.4 % |

**Table 2** Labelled features comparisons of ripe and unripe watermelons

| Mean | | Training set | | Testing set | |
|---|---|---|---|---|---|
| | | Ripe | Unripe | Ripe | Unripe |
| Features | | | | | |
| ZCR | | 0.0138 | 0.0202 | 0.0158 | 0.0175 |
| STE | | 5.6 | 8.2 | 7.4 | 7.8 |
| Sub-band STE ratio | 1 | 31.7 % | 66.9 % | 31.2 % | 57.4 % |
| | 2 | 29.0 % | 21.0 % | 27.6 % | 22.7 % |
| | 3 | 24.0 % | 5.95 % | 23.5 % | 13.7 % |
| | 4 | 15.6 % | 6.32 % | 19.0 % | 6.41 % |
| Brightness | | 12.8 | 17.5 | 23.8 | 13.4 |

3. *Sub-band Short-Time Energy Ratio* values of unripe watermelons drop much faster than those of ripe watermelons. This indicates that first sub-band acoustic signals from unripe watermelons contribute most to the *STE*, while for ripe watermelons each sub-band contributes similarly.
4. In the training dataset, the *brightness* of ripe watermelons is lower than that of unripe watermelons. However, this situation changed totally in the testing set, which indicates that *brightness* should not be adopt in the final feature vector.

### 6.3 Feature effectiveness

We measured accuracy, recall and precision of models constructed with each feature before combining all the features together. Let tp be the number of ripe watermelons that are classified as ripe, tn be the number of unripe watermelons that are classified as unripe, fp be the number of ripe watermelons that are classified as unripe and fn be the number of unripe watermelons that are classified as ripe. Then,

$$\text{Accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}} \tag{9}$$

$$\text{Recall} = \frac{\text{tp}}{\text{tp} + \text{fn}} \tag{10}$$

$$\text{Precision} = \frac{\text{tp}}{\text{tp} + \text{fp}} \tag{11}$$

Table 3 shows the accuracy of models constructed with each feature for ripe, unripe and overall. From the table, it can be seen that *first sub-band STE ratio* classifies ripe watermelons most accurately while the *fourth sub-band STE ratio* has the best performance for classifying unripe watermelons. The overall most accurate classifier is the *fourth sub-band STE ratio*. From the table, we can also see that classification result of *brightness* is irrelevant to watermelon ripeness.

**Table 3** Ripe, unripe and overall accuracy of each feature

| Accuracy | | Ripe (%) | Unripe (%) | Overall (%) |
|---|---|---|---|---|
| Classifier | | | | |
| ZCR | | 63.2 | 57.1 | 59.6 |
| STE | | 57.9 | 69.4 | 64.6 |
| Sub-band STE ratio | 1 | 84.2 | 81.6 | 82.7 |
| | 2 | 42.1 | 69.0 | 57.8 |
| | 3 | 84.2 | 55.1 | 67.2 |
| | 4 | 68.4 | 95.9 | 84.5 |
| Brightness | | 36.8 | 55.1 | 47.5 |

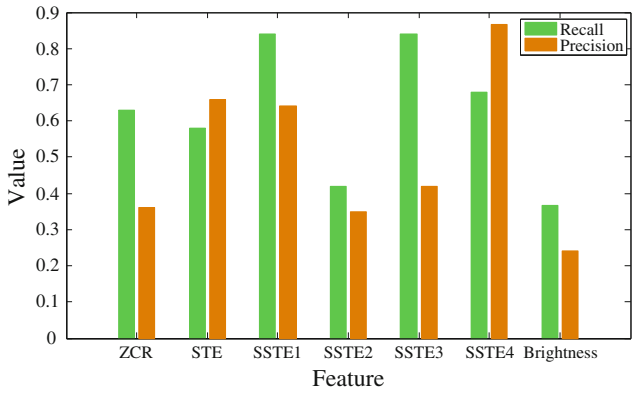


**Fig. 9** Precision and recall of each labelled feature

Figure 9 shows the precision and recall of each feature. Among the features, the first and fourth sub-band STE ratio stand out, while brightness performs worst.

### 6.4 Composite classifier

Normally, a combination of related features can increase the accuracy of a classification model. In our final classification model, we adopt features that are able to classify watermelon ripeness most accurately into a feature vector. *Brightness* is discarded in the final feature vector since its accuracy was found in the previous analysis to be low. Sub-band STE $\text{Ratio}_2$ is also discarded as its precision and recall are not satisfying. Hence, the final feature vector contains ZCR, STE, $\text{Sub} - \text{band STE Ratio}_i$, where $i \in \{1, 3, 4\}$. Final classification results show that the composite feature vector correctly classifies 89.3 % ripe, 90.4 % unripe and 89.9 % overall watermelons, with 84.2 % recall and 79.1 % precision.

## 7 Conclusions and future work

In this paper, we presented a method that uses mobile devices to record and analyse acoustic signals generated from thumping watermelons to classify their ripeness. We found watermelon ripeness-related acoustic features and combined these into a feature vector used to construct a classification model. The classification model was tested and found to be able to correctly classify watermelon ripeness. A real-time crowdsourcing application was implemented for Android and released on Google Play.

Currently, users can only give us feedback through emailing. This is not particularly convenient for us to collect and analyse data. Thus, in the future, we will aim to use a server that automatically analyses feedback sent by users to improve the classification model. Meanwhile, we will also study how other aspects of watermelons, such as shape and size affect the acoustic signals, and consider crowd-sourced use with other fruit.

Mobile devices detecting and analysing acoustic signals can also be applied to other domain, such as medicine. For example, by analysing the heartbeat of healthy people and heart disease patients, we may be able to develop a mobile application which helps heart disease patients monitor their condition. Mobile devices can help improve people's daily lives with more such applications. It is likely that many more such applications will be developed in future, accompanied by more research into improving the usability and performance of the foundational machine learning and crowdsourcing techniques.

**Acknowledgments** This work was established at the Singapore-ETH Centre for Global Environmental Sustainability (SEC), co-funded by the Singapore National Research Foundation (NRF) and ETH Zurich. We would also like to thank developers of open libraries like SVM and FFT used in our implementation. Without their work, we would spend much more time on implementing these algorithms.

## References

1. Aizerman A, Braverman E, Rozoner L (1964) Theoretical foundations of the potential function method in pattern recognition learning. Autom Remote Control 25:821–837
2. Ay C (1996) Acoustic evaluation of watermelon internal quality-maturity, cavity existence and orientation. J Agric Mech 5(4):57–71
3. Baki S, Annuar Mohd ZM, Yassin IM, Hasliza AH, Zabidi A (2010) Non-destructive classification of watermelon ripeness using Mel-frequency cepstrum coefficients and Multilayer Perceptrons. In: Proceedings of international joint conference on neural networks (IJCNN), IEEE, pp 1–6
4. Balandina E, Trossen D (2006) Nokia remote sensing platform middleware and demo application server: features and user interface. Nokia Research Center, Helsinki
5. Butterworth S (1930) On the theory of filter amplifiers. Wirel Engineer 7:536–541
6. Cantwell M (1996) Case study: quality assurance for melons. Perishables Handling Newsl Iss (85):10–12
7. Diezma-Iglesias B, Ruiz-Altisent M, Barreiro P (2004) Detection of internal quality in seedless watermelon by acoustic impulse response. Biosyst Eng 88(2):221–230
8. Gouyon F, Pachet F, Delerue O (2000) On the use of zero-crossing rate for an application of classification of percussive sounds. In: Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00), Verona, Italy. Helsiniki University of Technology, Helsiniki
9. Juillerat N, Müller Arisona S, Schubiger-Banz S (2007) Real-time, low latency audio processing in java. In: Proceedings of the international computer music conference, Copenhagen, Denmark
10. Lu H, Pan W, Lane ND, Choudhury T, Campbell AT (2009) Soundsense: scalable sound sensing for people-centric applications on mobile phones. In: Proceedings of the 7th international conference on mobile systems, applications, and services. ACM, Cumberland, pp 165–178
11. Lu L, Zhang HJ, Li SZ (2003) Content-based audio classification and segmentation by using support vector machines. Multimed Syst 8(6):482–492
12. McKinney MF, Breebaart J (2003) Features for audio and music classification. In: Proceedings of the third international symposium on music information retrieval (ISMIR), vol 3, pp 151–158
13. McLoughlin IV (2009) Applied speech and audio processing: with Matlab examples. Cambridge University Press, Cambridge
14. Misra A, Essl G, Rohs M (2008) Microphone as sensor in mobile phone performance. In: Proceedings of the international conference for new interfaces for musical expression (NIME-08), Genova, Italy
15. Mitrović D, Zeppelzauer M, Breiteneder C (2010) Features for content-based audio retrieval. Adv Comput 78:71–150
16. Pohjalainen J (2007) Methods of automatic audio content classification. Ph.D. thesis, Helsiniki University of Technology
17. Saunders J (1996) Real-time discrimination of broadcast speech/music. In: Proceedings of international conference on acoustics, speech, and signal processing (ICASSP), vol 2, IEEE, pp 993–996
18. Yamamoto H, Iwamoto M, Haginuma S (1980) Acoustic impulse response method for measuring natural frequency of intact fruits and preliminary applications to internal quality evaluation of apples and watermelons. J Texture Studies 11(2):117–136