# AUTOMASHUPPER: AN AUTOMATIC MULTI-SONG MASHUP SYSTEM

**Matthew E. P. Davies, Philippe Hamel, Kazuyoshi Yoshii and Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{matthew.davies, hamel.phil, k.yoshii, m.goto} [at] aist.go.jp

## ABSTRACT

This paper describes *AutoMashUpper*, an interactive system for creating music mashups by automatically selecting and mixing multiple songs together. Given a user-specified input song, the system first identifies the phrase-level structure and then estimates the "mashability" between each phrase section of the input and songs in the user's music collection. Mashability is calculated based on the harmonic similarity between beat synchronous chromagrams over a user-definable range of allowable key shifts and tempi. Once a match in the collection for a given section of the input song has been found, a pitch-shifting and time-stretching algorithm is used to harmonically and temporally align the sections, after which the loudness of the transformed section is modified to ensure a balanced mix. AutoMashUpper has a user interface to allow visualisation and manipulation of mashups. When creating a mashup, users can specify a list of songs to choose from, modify the mashability parameters and change the granularity of the phrase segmentation. Once created, users can also switch, add, or remove sections from the mashup to suit their taste. In this way, AutoMashUpper can assist users to actively create new music content by enabling and encouraging them to explore the mashup space.

## 1. INTRODUCTION

Mashups form a key part of the remix culture in music production and listening. Created by mixing together multiple songs, or elements within songs, music mashups hold strong potential for entertaining and surprising listeners by bringing together disparate musical elements in unexpected ways. Until recently, the process for creating mashups relied on two elements: first, the requisite musical imagination (and access to a large and varied music catalogue) to determine which songs to mix together, and second, the technical ability to use a Digital Audio Workstation to produce high quality results.

Due to the high popularity of mashups, some commercial systems and online tools have become available to assist users (both professional DJs and amateurs) in mixing songs and creating mashups. *DJ Mix Generator*[1] is an online database of 30,000 songs, where users can search by tempo, key and genre to find compatible songs to mix together. The *Harmonic Mixing Tool*[2] offers similar functionality, but instead of presenting results through an online search engine, it analyses a user's collection to identify song compatibility and can create a "harmonic fade" when mixing between songs. *Mashup*[3] also provides a harmonic compatibility measure between songs, which is determined using a key signature detection algorithm and relationships in the circle of fifths. To allow the manual creation of mashups, *Mashup* has an advanced audio editing interface.

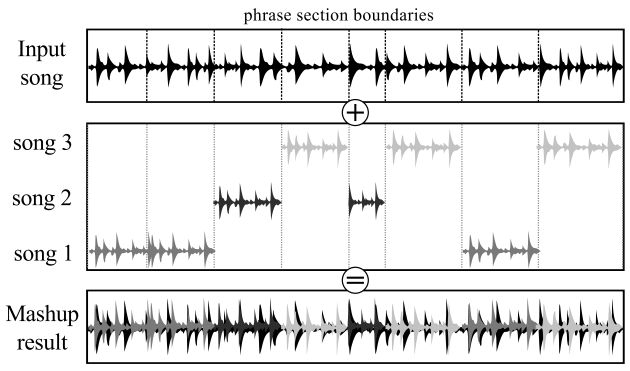
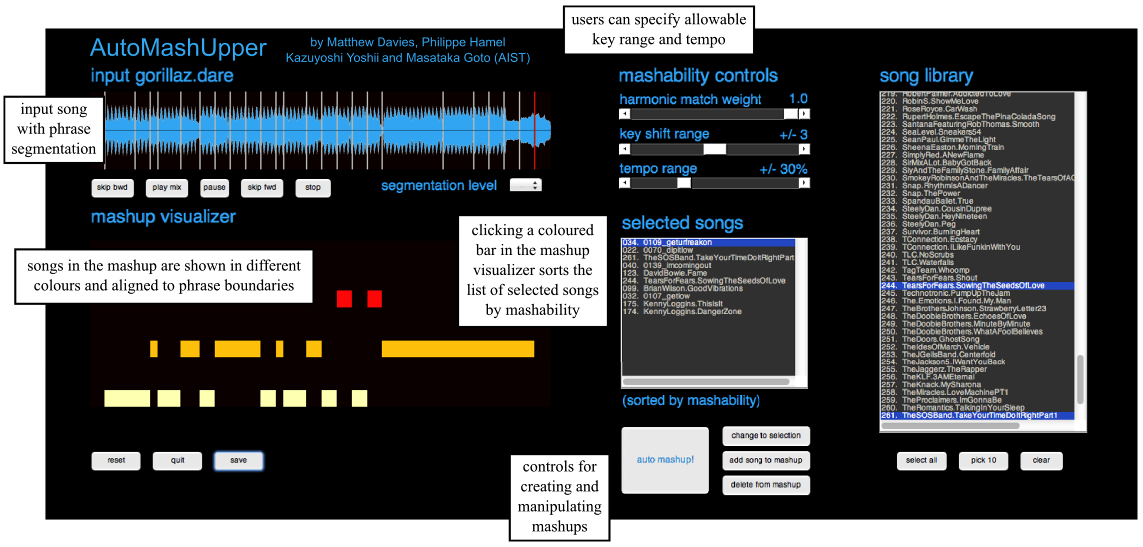


**Figure 1**. The concept of multi-song mashups.

Given the existence of these commercial mashup tools and their use of MIR techniques such as key detection, beat tracking and tempo estimation, it is quite surprising that so few research papers exist on this topic. Of those which do, their scope appears limited to using just a handful of musical excerpts, and they focus on the engineering aspects of time-stretching multiple songs simultaneously [7] or on visualisation as part of the mashup making process [12]. While these elements are certainly important, we believe that there are many opportunities for the development of MIR techniques within the field of music mashups. Indeed, mashup creation was recently highlighted as one of the "grand challenges" of MIR [5, p.222].

It is within this light that we propose *AutoMashUpper*, a system for making automatic multi-song mashups, as shown in Figure 1. The main novelty of our system lies

[1] http://www.djprince.no/site/DMG.aspx
[2] http://www.idmt.fraunhofer.de/en/Service_Offerings/technologies/e_h/harmonic_mixing_tool.html
[3] http://mashup.mixedinkey.com/


AutoMashUpper
by Matthew Davies, Philippe Hamel
Kazuyoshi Yoshii and Masataka Goto (AIST)
users can specify allowable key range and tempo
input song with phrase segmentation
skip bwd
play mix
pause
skip fwd
stop
mashup visualizer
songs in the mashup are shown in different colours and aligned to phrase boundaries
clicking a coloured bar in the mashup visualizer sorts the list of selected songs by mashability
mashability controls
selected songs
(sorted by mashability)
song library
reset
quit
save
controls for creating and manipulating mashups
auto mashup!
change to selection
add song to mashup
delete from mashup
select all
pick 10
clear

input gorillaz.dare

segmentation level

harmonic match weight 1.0

key shift range +/- 3

tempo range +/- 30%

selected songs:

034. 0109_geturfreakon
022. 0070_dipitlow
261. TheSOSBand.TakeYourTimeDoItRightPart
040. 0139_imcomingout
123. DavidBowie.Fame
244. TearsForFears.SowingTheSeedsOfLove
099. BrianWilson.GoodVibrations
032. 0107_getlow
175. KennyLoggins.ThisIsIt
174. KennyLoggins.DangerZone

song library:

219. RobertPalmer.AddictedToLove
220. RobinS.ShowMeLove
221. RoseRoyce.CarWash
222. RupertHolmes.EscapeThePinaColadaSong
223. SantanaFeaturingRobThomas.Smooth
224. SeaLevel.Sneakers54
225. SeanPaul.GimmeTheLight
226. SheenaEaston.MorningTrain
227. SimplyRed.ANewFlame
228. SirMixALot.BabyGotBack
229. SlyAndTheFamilyStone.FamilyAffair
230. SmokeyRobinsonAndTheMiracles.TheTearsOfAC
231. Snap.RhythmIsADancer
232. Snap.ThePower
233. SpandauBallet.True
234. SteelyDan.CousinDupree
235. SteelyDan.HeyNineteen
236. SteelyDan.Peg
237. Survivor.BurningHeart
238. TConnection.Ecstacy
239. TConnection.ILikeFunkinWithYou
240. TLC.NoScrubs
241. TLC.Waterfalls
242. TagTeam.Whoomp
243. TearsForFears.Shout
244. TearsForFears.SowingTheSeedsOfLove
245. Technotronic.PumpUpTheJam
246. The.Emotions.I.Found.My.Man
247. TheBrothersJohnson.StrawberryLetter23
248. TheDoobieBrothers.EchoesOfLove
249. TheDoobieBrothers.MinuteByMinute
250. TheDoobieBrothers.WhatAFoolBelieves
251. TheDoors.GhostSong
252. TheIdesOfMarch.Vehicle
253. TheJGeilsBand.Centerfold
254. TheJackson5.IWantYouBack
255. TheJaggerz.TheRapper
256. TheKLF.3AMEternal
257. TheKnack.MySharona
258. TheMiracles.LoveMachinePT1
259. TheProclaimers.ImGonnaBe
260. TheRomantics.TalkingInYourSleep
261. TheSOSBand.TakeYourTimeDoItRightPart1

**Figure 2**. Screenshot of the AutoMashUpper user interface. Additional descriptions of the functionality are overlaid.

in the estimation of what we term “mashability” - a measure of how well two songs fit, or mash together. Looking beyond the functionality of existing mashup systems which guide users to songs with matching key signatures and similar tempi, we incorporate a measure of harmonic similarity of beat synchronous chromagrams. This allows us to look for deeper matches than those possible from key-signature alone. Furthermore our measure of mashability can identify matches between songs in completely different key-signatures, by directly exploiting the knowledge that songs can be pitch-shifted by some number of semitones to “force” a match.

By identifying the phrase-level structure of a given input song, AutoMashUpper can determine the mashability between each phrase section of the input and songs in a user’s music collection. In this way, multiple songs can be used in the mashup at different times and this can radically increase the range and variety of possible mashups. To produce the mashup, we use existing techniques for pitch-shifting and time-stretching for harmonic and temporal alignment respectively, and loudness adjustment to create a balanced sound mixture.

In addition to the fully automatic mode, AutoMashUpper has a user interface to allow visualisation and manipulation of mashups, as shown in Figure 2. When creating a mashup, users can specify a list of songs to choose from, modify the mashability parameters and change the granularity of the phrase segmentation. Once created, users can also switch, add, or remove sections from the mashup to suit their taste, and additionally save the results for re-use.

The remainder of this paper is structured as follows. In Section 2 we describe the phrase-level segmentation used to partition songs in AutoMashUpper. In Section 3 we present our method for mashability estimation and describe how we produce the mashups. We then present the interface of AutoMashUpper in Section 4 and illustrate its main modes of operation. Finally, in Section 5, we discuss the potential impact of our system, towards motivating further research into mashup generation, and present some areas for future work.

## 2. PHRASE-LEVEL SEGMENTATION

A central component of our mashup system, and the key to enabling multiple songs to be used to produce the musical result, is a phrase-level segmentation of the input. While much research has been conducted into structural segmentation of music signals (e.g., [11, 13]) their goal is to identify boundaries of long sections corresponding to intro, verse and chorus, and to apply labels to these sections to identify repetitions. For our purpose in creating mashups, we require a similar type of analysis, however our concern is not directly in labelling the sections, but rather in precisely identifying temporal boundaries. Furthermore we wish to identify shorter sections corresponding to musical phrases, rather than longer time scale structure such as verse or chorus.

Through informal experimentation with existing segmentation algorithms with publicly available implementations (e.g., [13]) we discovered that it was not trivial to reliably sub-divide the estimated sections into downbeat synchronous phrases. On this basis, and in the interest of avoiding multiple separate stages of processing the input signal, we devise our own method for phrase segmentation, adapting elements from existing systems to suit our needs. Since an important element of mashups is the harmonic compatibility of the mixed music signals, we base our phrase-level segmentation on a harmonic representation of the input.

To generate the harmonic signal representations for phrase-level segmentation and the subsequent estimation of mashability, we use the NNLS Chroma plugin [8] within Sonic Annotator [2]. Given an input audio signal, we extract three results from the NNLS Chroma plugin: the global tuning, $t$, of the input, an 84-bin (7 octave), tuned semitone spectrogram, $S$, and a 12-dimensional chromagram (the distribution of energy across the chromatic pitch classes in a musical octave), $C$. All outputs are extracted using the default parameters. To create beat-synchronous versions of $S$ and $C$, we use the QM Vamp beat tracking plugin in Sonic Annotator, and take the median across the time frames per beat. For ease of notation, we will continue to use $S$ and $C$ to refer to the beat-synchronous versions.

To simplify our approach, we make the following assumptions about the songs to be used for mashups: all phrase sections are a whole number of measures, all songs have a constant 4/4 time-signature, and the input tempo is approximately constant.

To determine the phrase boundaries, we group the beat-synchronous frames of $S$ into measures, to create a downbeat-synchronous semitone spectrogram. To identify the downbeats we used a modified version of the method by Davies and Plumbley [3] with $S$ as the main input. As shown in [13] it can be beneficial for segmentation performance to "stack" beat frames together when estimating section boundaries. In this way, we group sets of four consecutive beat frames (starting at each downbeat and without overlap) to create a downbeat-synchronous stacked semitone spectrogram.

Given the beats and downbeats, we then follow the classical approach of Foote [4] for structural segmentation. We calculate a self-similarity matrix from the downbeat-synchronous stacked semitone spectrogram using the Itakura-Saito distance [6] and slide a Gaussian checkerboard kernel along the main diagonal to generate a novelty function to emphasise section boundaries. As shown in [4] the size of this kernel has a direct impact on the level of the segmentation and temporal precision of the boundaries. Since our interest is in finding short phrase-level sections, we use a small kernel of size eight downbeats. To obtain an initial set of phrase boundaries we peak-pick the resulting novelty function. We then employ a technique derived from [11] to maximise the regularity of the detected phrase boundaries. A graphical example is shown in Figure 3 along with a flow chart in Figure 4(a).

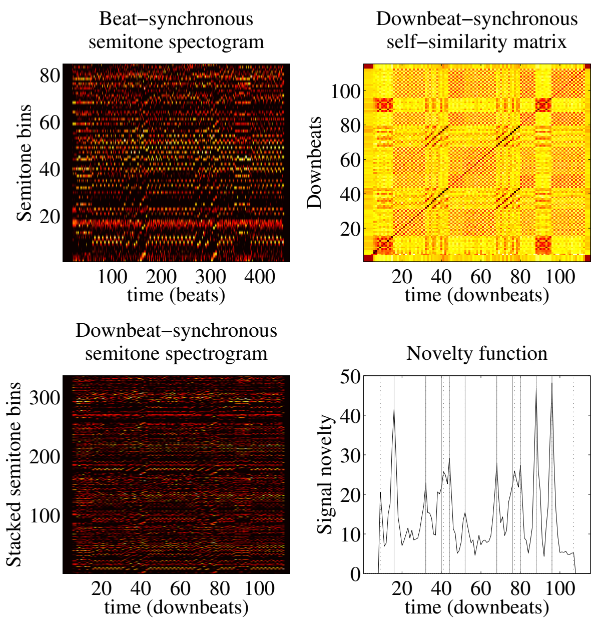


**Figure 3**. Phrase-level segmentation overview. (top left) a beat synchronous tuned semitone spectrogram. (bottom left) a downbeat-synchronous spectrogram, where groups of four beat frames are stacked into measures. (top right) a self-similarity matrix generated from the downbeat-synchronous semitone spectrogram. (bottom right) a novelty function whose peaks highlight likely phrase boundaries. The vertical dotted lines show raw phrase boundaries and the solid grey lines show the result of regularity-constrained realignment.

## 3. MAKING MASHUPS

This section describes how the mashability is estimated between beat-synchronous chromagrams for each phrase section of the input song, and the songs in a music collection. Then we address the requisite processing to physically create the mashup itself. A graphical overview of the complete mashup creation process is shown in Figure 4.

### 3.1 Estimating Mashability

Once the set of phrase segment boundaries has been determined, we turn our attention to finding a match for each phrase section of the input with songs in the users' music collection by estimating what we refer to as "mashability". For each song in the collection, we pre-calculate a beat-synchronous chromagram using the techniques described in Section 2 prior to estimating the mashability.

In contrast to existing systems which guide users towards mixing songs with matching key signature and have similar tempi, we argue that there is a much wider scope of potential matches (and potentially more interesting musical results) by considering mashups between songs in different keys and tempi. In effect, our approach is not only to look for matches according to the existing properties of songs, but also to look for matches in a kind of "transform" domain, in the knowledge that we can subsequently use time-stretching to temporally align songs, and pitch-shifting (by some number of semitones) to create, or indeed "force" a harmonic alignment.

We base our estimation of mashability around the harmonic similarity between beat-synchronous chromagrams. For the current phrase-section $p$ of length $K$ beats from the input song, $i$, we isolate the beat-synchronous chromagram $C_{i,p}$ (a 12-by-K matrix). To facilitate the search across different key shifts, we rotate the chroma bins of $C_{i,p}$ across a range of integer semitone shifts, $r$, which can be set from 0 to $\pm 6$ semitones according to user preference. For each key-shifted chroma section of the input, $C_{i,p,r}$ we measure its harmonic similarity across each rotational shift, $r$ to all possible beat increments $k$, (for $K$-beat frame chromagrams) for each song $n$ in the user's song collection using the Cosine similarity,

$$H_n(r,k) = \frac{C_{i,p,r} \cdot C_{n,p,k}}{||C_{i,p,r}||\ ||C_{n,p,k}||} \tag{1}$$

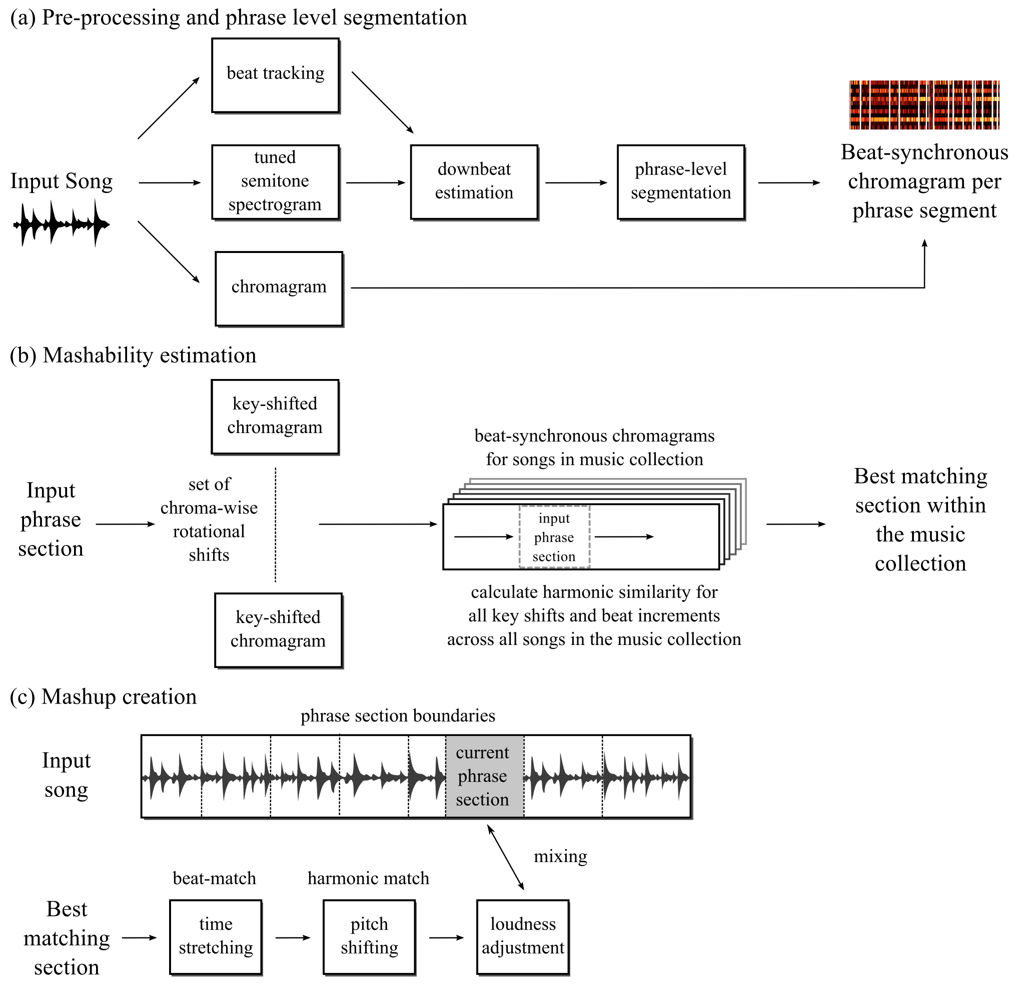

(a) Pre-processing and phrase level segmentation
beat tracking
Input Song
tuned semitone spectrogram
downbeat estimation
phrase-level segmentation
Beat-synchronous chromagram per phrase segment
chromagram
(b) Mashability estimation
key-shifted chromagram
Input phrase section
set of chroma-wise rotational shifts
beat-synchronous chromagrams for songs in music collection
input phrase section
Best matching section within the music collection
key-shifted chromagram
calculate harmonic similarity for all key shifts and beat increments across all songs in the music collection
(c) Mashup creation
phrase section boundaries
Input song
current phrase section
mixing
beat-match
harmonic match
Best matching section
time stretching
pitch shifting
loudness adjustment

**Figure 4**. Overview of the mashup creation process. (a) Pre-processing and phrase level segmentation, (b) mashability estimation, and (c) mashup creation.

where high harmonic similarity will have $H$ close to unity and low similarity will have $H$ close to zero. A graphical overview is shown in Figure 4(b).

To move from harmonic similarity to mashability, $M$, we include an additional term which rewards songs whose tempo, $T_n$ is within a user specified ratio, $\eta$, of the input tempo, $T_i$, such that:

$$M_n(r,k) = \begin{cases} H_n(r,k) + \gamma, & \text{if } |1 - |(T_i/T_n)|| \leq \eta \\ H_n(r,k), & \text{otherwise} \end{cases} \tag{2}$$

where $\gamma$=0.2 was found to give favourable results. Note that the greater the value of $\eta$ the more permissive the system in terms of allowable tempo matches between the input and songs in the collection.

Once the mashability has been calculated across the song collection we find the song, $n$, beat increment, $k$, (i.e. starting point) and rotational shift, $r$, which lead to the highest mashability for the current input phrase section, $C_{i,p,r}$. We apply the rotations of chroma to the *input* chromagram and not to the songs in the database, which remain unaltered by the search across mashability space. Therefore, when we come to implement any required key-shift to match the selected song with the input, we must pitch-shift the selected song by $-1 * r$ semitones.

By measuring the harmonic similarity across all beat increments $k$ and rotational shifts $r$ we create a large search space, which in turn gives the highest possibility for finding regions of high harmonic similarity. Furthermore we have found matching between chroma matrices at incremental beat shifts, rather than looking at individual chroma frames, we can implicitly capture aligned chord changes between songs – a factor we have found improves the quality of the resulting mashup.

## 3.2 Mashup Creation

The final part of the automatic mashup creation process is to transform the selected section and mix it with the input, as shown in Figure 4(c).

To create this mix, several steps are required. First, we use the open-source Rubberband time-stretching and pitch-shifting library [4] to temporally align (or "beat-match") the matching section with the current phrase section of the input song. This is achieved using the `mapfile` function (within Rubberband) which specifies a set of anchor points, i.e., the beats of the song to be transformed, and a cor-

[4] `http://breakfastquay.com/rubberband/`

responding set of target times – the beats of the current phrase section of the input.

Once aligned in time, the matching section is then harmonically aligned to the input phrase using the pitch-shifting functionality of Rubberband. This harmonic alignment addresses two areas: pitch-shifting by the required number of semitones, $r$, to match the key-signature of the songs, and a tuning correction, identified as the ratio between the estimated tuning for the input song and selected matching song. In the event that both a tuning and pitch correction are required, we combine these factors into a single processing call to Rubberband. In the event that the two songs are already matched in key, (i.e. $r$=0) and the difference in tuning is less than 1Hz, then the mashup can be made by beat-matching alone.

The final stage in mixing the sections of the two songs together, is to address any imbalance in the loudness between the current input section and the transformed match. To this end, we estimate the perceptual loudness in the input phrase section and transformed signal using the Replay Gain algorithm [10]. While traditionally used to equalise loudness between songs, we wish to give greater prominence in the mix to the input song, hence we scale the amplitude of transformed section to have 90% of the loudness of the input phrase section.

## 4. AUTOMASHUPPER

Up to this point we have described the backend processing to enable the *automatic* creation of mashups. To allow users of AutoMashUpper to be involved in the mashup creation process, we have built a user interface, which is shown in Figure 2. To illustrate the functionality of the user interface and to provide sound examples, demonstration videos are available here [5] .

### 4.1 Interface Overview

The layout of the interface is split into three sections. On the left hand side there are two main panels, the top for visualising the waveform of the input song and the estimated phrase level section boundaries. Below this is the mashup visualizer which shows the songs currently used in the mashup. In addition, a set of playback controls are included for listening to the input song and the mashup. On the right hand side is a panel containing the list of pre-analysed songs in the music collection. Below this listbox are buttons to select songs from the library to use in the mashup. In the central panel we have a listbox which shows the current songs selected for use in the mashup, as well a set of sliders for manipulating the parameters of the mashabilty calculation. These specify the range of allowable key-shifts, and the preferred tempo range. Beneath the list box are buttons for creating the automatic mashup and then subsequent manipulation of the result. This functionality is described in the following subsection.

[5] `http://www.youtube.com/user/automashupper`

### 4.2 User interaction

The typical scenario we envisage for AutoMashUpper is as follows. The user loads a song of their choice into the system, after which a waveform of the input song appears in the top left panel along with vertical bars to indicate the estimated phrase section boundaries. The user can listen to the input song and click on different sections for the playback to jump directly to these parts of the song. In addition the user can explore finer segmentations where phrase sections can be sub-divided into 16, 8 or 4-beat units using the segmentation level drop-down menu. Having selected a segmentation level, the user can then choose a set of songs from the song library on the right hand side of the interface. For this, three options are available: i) to manually select a sub-set of their choice; ii) to select all of the songs in the library; or iii) to pick ten random songs.

When manually choosing a subset, we have found that only picking songs from the same artist, or the same album, i.e., *artist-level-mashups* or *album-level-mashups*, can lead to very pleasing results due to high timbral compatibility.

The songs chosen by the user then appear in the listbox of selected songs in the middle of the interface. Using the sliders above this listbox, the user can specify how wide a range of key shifts and tempi to allow in the mashability estimation. Specifying a small range of key shifts and tempi can lead to somewhat conservative results, whereas allowing a wide range of possibilities in the mashup space can facilitate better matches, but perhaps at the cost of creating more unusual results, for example where a transformed song could be pitch-shifted up or down by five semitones or radically changed in speed.

Once AutoMashUpper has been parameterised, the user can then hit the **auto mashup!** button to generate a mashup. Or, the user may simply hit this button right after loading the input song. As each section is identified and added to the mashup it appears in the lower left hand panel, where each song is displayed in a different colour. When the processing has finished the user can listen to the result – once again with the ability to navigate between phrase sections by clicking in the appropriate region of the waveform representation or the mashup visualizer panel. During playback, a red vertical line indicates the currently playing phrase section of the input song.

Clicking a particular bar in the mashup visualizer will highlight the name of the chosen song in the selected songs listbox. It will also re-order the remaining songs in descending order of mashability. At this point the user can make a subjective judgement over whether they like the mashup as it is or which to change it. The user has three options: first, they can delete the currently used section from the mashup, second, they can choose a different song from the selected songs listbox to replace it or third, they can choose to add another song from the list to the mashup. If the user is pleased with the resulting mashup they can use save button, which will create time-stamped .wav files for the input song, the generated mashup by itself and the mixture of the input and mashup. In addition it records a screenshot of the interface to show the list of songs used and mashability parameters.

## 5. DISCUSSION

In this paper we have presented *AutoMashUpper*, a system for the creation of multi-song mashups. Our main contribution in this work is a method for mashability estimation which enables the automatic creation of music mashups. Our work forms part of the emerging field of creative-MIR, where music analysis and transformation techniques are used within real applications towards the transfer of knowledge outside the MIR research community. We have designed AutoMashUpper with the aim of assisting users (who might lack music composition skills) to become music creators through simple interactions with a user interface. Our hope is that AutoMashUpper will encourage users to explore a wide space of mashup possibilities by manipulation of the mashability parameters, perhaps even creating a new genre of "auto" mashups.

We believe a particular advantage of the automatic approach to searching for mashability within a large collections of songs is that such a system can uncover musical relationships which might otherwise never be found. This is especially relevant if we consider the size of the search space when allowing for matches at the phrase-level of songs. Our current system uses a catalogue of around 300 songs from which we have been able to create many interesting mashups, with minimal effort. Furthermore, even when the phrase-level segmentation has errors, this has the potential to create unusual and unexpected results.

We have been particularly surprised by the quality of results achieved when using the "pick ten random songs" option in AutoMashUpper. This indicates that many hidden relationships exist between different sections of songs, and discovering them in the context of a music mashup appears a particularly good way to enjoy them. In this sense, the possibilities when applying this system to a very large music collection could be almost endless. However, the transition from a medium-sized collection to a very large one presents many challenges due to scalability and increased computational cost, and would require a much faster search technique, (e.g., [1]). We intend to explore this area within our future work as well as investigating source separation methods (e.g., [9]) to offer users even greater mashup creation possibilities.

Since mashups, by definition, contain multiple songs playing at once, they represent an interesting category of music from an auditory scene analysis perspective, where it is listeners' familiarity with songs in the mashup which allow them to understand a musical scene which might otherwise be too complex to process and hence appreciate [5]. To further explore these ideas and to address the lack of a formal evaluation of AutoMashUpper, we plan to undertake subjective listening tests to explore listeners' levels of musical engagement and understanding of mashups.

Looking beyond the current version of AutoMashUpper, we recognise that mashability is not fully explained by harmonic similarity alone, and we can envisage many additional uses of MIR techniques for creating more sophisticated measures of mashability, e.g. by exploring rhythmic and spectral compatibility. On this basis we strongly encourage other researchers to explore mashup creation methods to expand the field of creative MIR.

## 6. ACKNOWLEDGMENTS

This work was supported by OngaCREST, CREST, JST.

## 7. REFERENCES

[1] T. Bertin-Mahieux and D. Ellis. Large-scale cover song recognition using the 2D Fourier transform magnitude. In *Proceedings of 13th International Society for Music Information Retrieval Conference*, pages 241–246, 2012.

[2] C. Cannam, M. O. Jewell, C. Rhodes, M. Sandler, and M. d'Inverno. Linked data and you: Bringing music research software into the semantic web. *Journal of New Music Research*, 39(4):313–325, 2010.

[3] M. E. P. Davies and M. D. Plumbley. A spectral difference approach to extracting downbeats in musical audio. In *Proceedings of the 14th European Signal Processing Conference (EUSIPCO)*, 2006.

[4] J. Foote. Automatic audio segmentation using a measure of audio novelty. In *Proceedings of IEEE International conference on multimedia and expo*, pages 452–455, 2000.

[5] M. Goto. Grand challenges in music information research. In M. Muller, M. Goto, and M. Schedl, editors, *Multimodal Music Processing*, pages 217–225. Dagstuhl Publishing, 2012.

[6] F. Itakura and S. Saito. Analysis synthesis telephony based on the maximum likelihood method. In *Proceedings of the International Congress on Acoustics*, pages 17–20, 1968.

[7] G. Griffin Y. E. Kim and D. Turnbull. Beat-sync-mash-coder: A web application for real-time creation of beat-synchronous music mashups. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 437–440, 2010.

[8] M. Mauch and S. Dixon. Approximate note transcription for the improved identification of difficult chords. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, pages 135–140, 2010.

[9] Z. Rafii and B. Pardo. REpeating Pattern Extraction Technique (REPET): A simple method for music/voice separation. *IEEE Transactions on Audio, Speech and Language Processing*, 21(1):71–82, 2013.

[10] D. Robinson. *Perceptual model for assessment of coded audio*. PhD thesis, Department of Electronic Systems Engineering, University of Essex, 2002.

[11] G. Sargent, F. Bimbot, and E. Vincent. A regularity-constrained viterbi algorithm and its application to the structural segmentation of songs. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 483–488, 2011.

[12] N. Tokui. Massh!: A web-based collective music mashup system. In *Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts*, pages 526–527, 2008.

[13] R. J. Weiss and J. P. Bello. Identifying repeated patterns in music using sparse convolutive non-negative matrix factorization. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, pages 123–128, 2010.