University of Manchester
School of Computer Science
Project Report 2023

Automatic Drum Transcription

Author: Jack Hygate

Supervisor: Kung-kiu Lau

Abstract

Automatic Drum Transcription

Author: Jack Hygate

In this report we propose a method for automatic drum kit transcription by analysing video recordings of drummers playing with marked drumsticks. Drum music transcription is an essential tool in music education as it provides a systematic representation of performances that can be used for learning and practice. However, manual transcription is time-consuming and requires specialized knowledge, as such an accessible transcription method would be valuable for drummers. Using computer vision techniques drum elements (sticks and hi-hat position) are tracked, and drum hits are calculated. This data is then translated to both MIDI and drum notation. This approach is successful in transcribing performances and offers advantages over traditional audio-based methods in its ability to capture nuances in playing techniques such as stickings (which hand was used to play) and accents (if one drum was hit harder than the others).

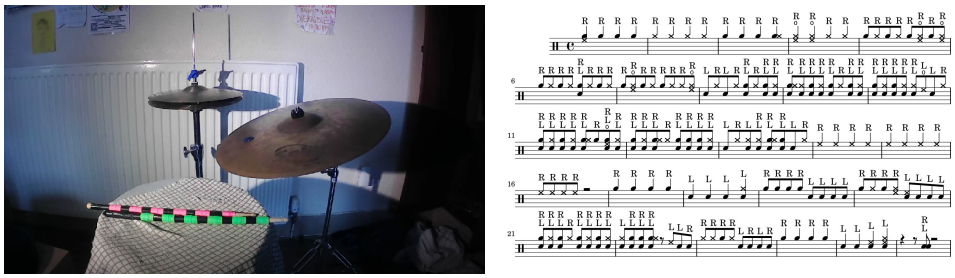
Supervisor: Kung-kiu Lau

Contents

Chapter 1

Introduction

Automatic music transcription is a process that aims to convert musical performances captured in audio or video recordings into symbolic notations, such as sheet music or MIDI files. This technology has significant applications in various fields, including music education, music analysis, and the preservation and sharing of musical works. The focus of this project is on the transcription of drum notation from video recordings, which involves the extraction of drumming information, such as the movement of drumsticks and hi-hats, as well as detecting the timing and type of drum hits. By leveraging computer vision techniques, the goal is to develop an automated system capable of accurately transcribing drum performances from videos, enabling musicians and enthusiasts to access and study drum music more efficiently and conveniently.



(a) Example Video Frame (b) Sheet Music

Figure 1.1: Sheet music generated from a video

1.1 Sheet Music

Drum sheet music, also known as drum notation, is a way to represent drum kit parts in written form. It uses a system of symbols and notation unique to percussion instruments, allowing musicians to understand and play the rhythm and patterns of a drum piece. In this section, we will provide a brief overview of drum sheet music and highlight key elements to help readers gain a better understanding of its structure and components.

Each symbol used in drum notation conveys two main pieces of information: the specific drum hit and when it was it. The following figures show the common notation used to represent each drum on the kit.

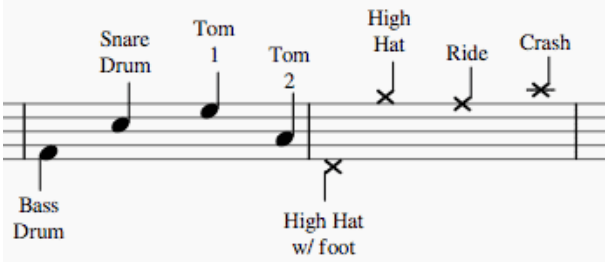


Figure 1.2: A drum key, showing the symbols for each drum

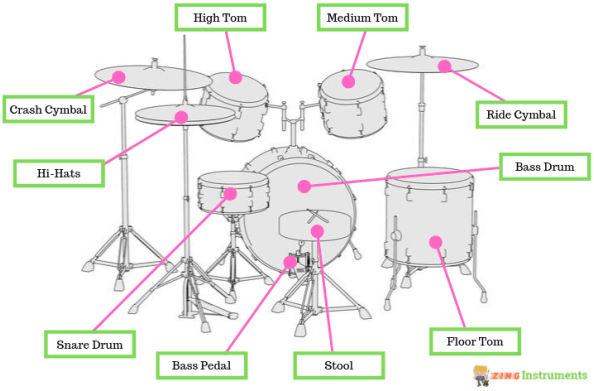


Figure 1.3: A labelled drum kit, showing the name of each drum

The rhythm and timing of the given hit is also conveyed by the notation. A detailed description of how this timing is read and played will not be given but the symbols used are shown below. The important information to note is that a specific symbol is needed to show when a note is played and that rests are used. A rest is a symbol that tells a drummer not to play at a specific time, the rest symbol has several variations that are used which depend on its local context, e.g. the rhythm of the notes recorded previously.

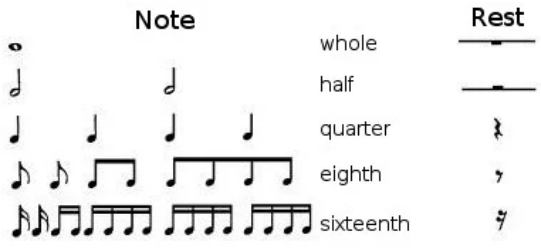


Figure 1.4: The symbols used to show rhythm and timing

Another concept of drum notation is recording whether the hihat is open or shut when the hihat is hit as this drastically changes the sound produced. An open hihat is described by a small circle above the hihat notation as shown below.

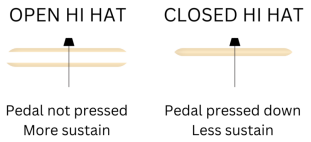


Figure 1.5: An open and shut Hihat

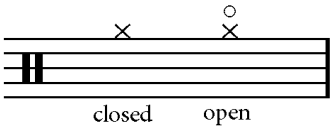


Figure 1.6: The symbols used to show when a hihat is open or closed

The intensity of a drum hit can be recorded in sheet music as an accented note. An accented hit tells the reader to play that drum louder than the unaccented notes in the piece.

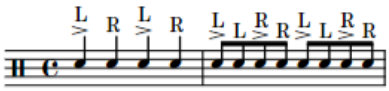


Figure 1.7: Accented notes

One final concept that will be relevant to this project is sticking - simple notation used to indicate which hand the note is played with. While this notation does not change the sound of the groove played, it is useful information to convey to the reader how best to play the groove.

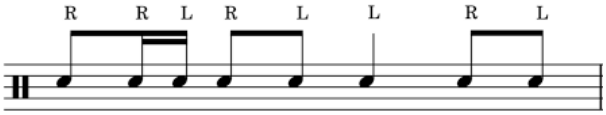


Figure 1.8: A simple snare groove with stickings shown

All of these features of a drum performance must be captured and recorded in the sheet music produced.

1.2 Motivation

The aim of the project is to automatically generate both a MIDI representation of a drum performance, and a drum notation representation. Both of these formats are very useful:

Drum notation is a useful tool for several reasons:

- **Communication** - Drum sheet music provides a standardized and universally understood representation of drum patterns, rhythms, and techniques. This allows drummers of different skill levels and backgrounds to communicate and share ideas effectively using a

common language. Furthermore, sheet music is essential in collaborative projects such as bands, orchestras and ensembles so that all musicians are on the same page.

- **Learning and practice** - Sheet music allows a drummer to quickly learn a given song and practice exercises. Many practice books are available for students to practice rudiments (key techniques that are building blocks for drummers). These exercises are often pages of simple rhythms with stickings shown. An example is shown in figure 1.9. The ability to generate these simple rudiment exercises quickly would be very useful.
- **Preservation and Documentation** - Recording a performance allows it to be persevered for further analysis in a format that is more accessible than viewing a video or listening to audio.

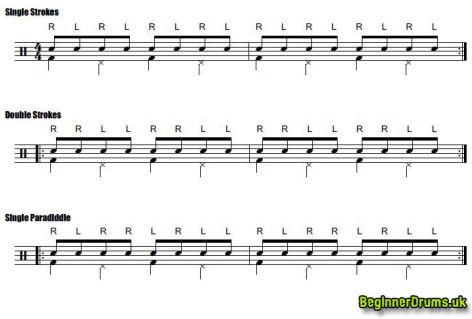


Figure 1.9: Example rudiment exercises

A MIDI representation of a drum performance is also very useful:

- **Electronic Music** - Many artists create music via software with virtual MIDI instruments. This is done via having a MIDI input instrument e.g. an electronic kit or using software to design a groove via a UI. This means that creating electronic drums is less accessible to drummers without an electronic kit as they cannot generate an electronic groove with only an acoustic kit and other tools would have to be used. This project allows a user to produce a MIDI file to be used in a project with only their acoustic kit and a camera.
- **Editing a groove** - After the user has generated a MIDI groove they can then use free to use music editing software to manipulate their performance. Mistakes can be fixed, tempo can be changed and the drum sounds can be replaced with any other sound.
- **MIDI instrument** - A MIDI drumkit allows custom sounds to be selected and played live on a drum hit. This functionality is usually served by an electronic kit or a sample pad, both of which are relatively expensive. By using the a camera to capture drum hits in real time, an acoustic kit can be used as a MIDI instrument to play electronic sounds live by using the software as a MIDI instrument,

Whilst these drum representations are useful to have access to, the process of transcription is time consuming and requires skill in understanding sheet music. Because of this the quantity and quality of drum sheet music is low, with many drummers tending to play by ear e.g. attempt to learn songs by listening to them. This means that the process of learning a song or a groove is slower than it could be for some players.

There are transcription services that exists but these are not accessible; professionals offer their services for a fee and some audio based ML models are available (which require training on the users specific kit sounds) [21].

Because of these reasons it is clear that accessible software that allows a user to transcribe their playing and generate MIDI representations would be a useful tool.

1.3 Aims and Objectives

The main aim of this project is to: **Generate both MIDI and drum notation of a drum performance from a video by tracking marked drumsticks**. This can be split into the following objectives that can be evaluated.

Track two fast-moving blurry marked drumsticks

1. Detect a stationary drum stick marked with coloured markers
2. Track a fast moving stick that suffers from motion blur

These goals will be evaluated by determining stick tracking accuracy as the number of frames a sticks position is correctly calculated. The stick tracking accuracy will be calculated for varying conditions.

Detect when a specific drum is hit and how

1. Detect whether the hi-hat is open or closed
2. Detect whether a hit is an accent

Each of these subgoals will be evaluated by finding their accuracy against ground truth.

Generate MIDI and drum notation representations

1. Produce correct sheet music that records the groove played
2. Produce a MIDI file that corresponds to the groove played

The final results generated can be evaluated by comparing to manual transcriptions of the same performances. Furthermore the MIDI file can be played over the performance to detect obvious errors.

Chapter 2

Background and Challenges

2.1 MIDI

Introduction to MIDI

MIDI (Musical Instrument Digital Interface) is a standard protocol [5] for communicating musical information between digital devices, such as computers, synthesizers, and drum machines. MIDI files store musical data as a series of messages that represent various events, such as note on/off, pitch bend, and tempo changes. Unlike audio files, MIDI files do not contain actual sound; instead, they contain information about how a device should play the music. This makes MIDI files extremely versatile and lightweight, as they can be used to control different instruments and sounds based on the user's preference.

MIDI for Drums

When it comes to drums, MIDI files are used to represent the individual drum hits and their respective timing. Each drum sound or instrument is assigned to a specific MIDI note number, which is determined by the General MIDI (GM) standard. For example, a kick drum might be represented by note number 36, while a snare drum might be represented by note number 38. These note numbers are consistent across various devices and software, allowing for seamless communication and compatibility.

Creating MIDI Files for Drums

To create a MIDI file for drums, the following steps can be followed:

1. Analyze the drum performance: In the case of a drum transcription system, the input can be a video or audio recording of a drum performance. The system processes the input to detect drum hits, their timing, and the corresponding drum types (e.g., kick, snare, hi-hat, etc.).

2. Map drum hits to MIDI notes: Once the drum hits are identified, they need to be mapped to their corresponding MIDI note numbers based on the General MIDI standard. This step essentially translates the drum performance into a series of MIDI note on/off events.

3. Add additional MIDI information: Along with note on/off events, other MIDI information, such as velocity (how hard a note is struck) and timing (e.g., tempo, time signature), can be added to the MIDI file to enhance its expressiveness and accuracy.

4. Save the MIDI data: Finally, the MIDI data is saved in a file format, typically .mid, which can be used by various devices and software for playback or further editing.

2.2 Related Research

Automatic drum transcription (ADT) is a well researched topic from the perspective of audio analysis. Many papers have been written about techniques that can transcribe a drummers performance from the audio.

2.2.1 "A Review of Automatic Drum Transcription"

[21] An in depth review of ADT is given in this paper. It provides a comprehensive review of ADT research, defines key challenges, and discusses the major drawbacks of current methods. Simple methods are described, such as characteristic band pass filtering [12], this is a simplistic method that uses a gradient decent function to define a frequency range that defines each key drum sound. More complex methods are described using complex drum sound feature representations (such as spectral flux, zero crossing rate and learned features) [16].

A relevant paper discussed is "Automatic transcription of drum sequences using audiovisual features" [11]. It tests the hypothesis that audio features can enhance transcription accuracy using a support vector machine classifier trained on audio and visual features. The video features are separated into movement of the drummer and sticks - gesture masks , and movement of the drums - drum masks. To define these masks, the drum areas must be user selected. These are valuable features as a learned representation or a drum vibration can be found as well as a learned representation of drum stick movements. The paper shows that visual features on their own result in lower transcription than using purely audio based features. (64.0% vs 81.5%). A combination of audio and visual features only increases accuracy a little (to 85.8%). It seems that the visual features may be suboptimal as they include a lot of noise, the movement of the drummer and vibrations of drums that are not due to a direct hit. It is unclear if cleaner features, such as the exact drumstick positions, would result in stronger visual transcription accuracy.

The key takeaways from this review are the drawbacks of audio methods:

1. **Reliance on training data** - Drums come in a huge variety. The sounds one snare drum might make are dissimilar from another snare drum. For audio ADT, the classifiers must be trained on drum sounds from the kit being used This means it would be difficult and time consuming for a user to generate a set of training data for their drums.

2. **Loss of drum nuance** - Key drum information such as stickings and accents are not recorded in audio based ADT. In fact without visual features, sticking information is impossible to extract.

3. **Sensitive to background noise** - Drum event onset detection is subject to background noise. This is particularly an issue if a drummer is drumming with other instruments. It seems likely that visual data will help to ignore other instruments.

2.2.2 AeroDrums

AeroDrums[1] are a company who use a similar approach to the one proposed in this report to play drums without a drumkit, producing sound when a user is playing ”air drums”. They use a retrofrelective marker on the end of each stick, an infrared light source, and a camera placed directly in front of them, to capture the position of the end of sticks. As there is no drum kit in front of them to occlude the sticks, they produce accurate results for hitting imaginary drums and can even be use to produce sheet music. The major drawback to this solution is that real drums cannot be used at the same time. This means that the intricacies of drumming are lost, a drummer cannot use techniques that rely on bouncing on a physical drum (e.g rolls) so nuance is lost. Furthermore, this solution cannot be used to record a real life performance on actual drums. However this product demonstrates that tracked drumsticks can be used to generate accurate transcriptions.

2.2.3 Freedrum

Freedrum [3] is another drum transcription company. For $349.00 they offer a setup including 2 drumsticks and 2 pedals with inbuilt sensors and an associated app. This System is certainly not accessible due to the price. Exact 3D positioning of each drumstick can be tracked with this setup resulting in very accurate transcription. However this solution seems unnecessary, the same information should be able to be extracted from audio and visual cues, without needing to resort to expensive, exact positioning sensors. This product does demonstrate that a cheaper transcription service would be valuable.

2.2.4 Drumstik

Drumstick [2] is a pure audio transcription service. It is capable of producing accurate results at high speeds by listening to just audio. However it is limited to just 3 drums (hihat, snare and bass drum). It also suffers from the same issues as the audio ADT methods described earlier. However, by only transcribing the three core drums, training for a specific kit is unnecessary as bass, snare and hihat representations can be generalised.

2.3 Challenges

After analysing the problem, many key challenges were found that must be overcome to produce reliable results.

2.3.1 Stick Detection

Marker Detection

The first step in stick detection and tracking is detecting the individual markers that are being tracked. This involves describing the colour of markers to be tracked, and detecting them, taking into account any changes in colour that may occur from changes in illumination and motion blur. A reliable method of detecting a coloured markers is by using a HSV colour range. Depending on the specific frame where colours are detected, multiple markers may merge

together, a single marker may be split into two, or random noise may be detected as a marker. Morphological operations that help combat these issues are described in subsection 2.4.3.

Marker Misdetection

As all stick markers lie on a line, it is simpler to overdetect markers and remove outliers, than to underdetect and miss key information about stick position. Also, regions of colour that correspond to a marker colour may be visible in in the frame, e.g on the drummers clothing or in the background. Because of this a method for removing outliers and determining which colour regions correspond to stick markers is key. A technique for achieving this (RANSAC) is described in subsection 2.4.4.

2.3.2 Stick Tracking

High speeds, frame rate issues

Drum sticks tend to move very fast [9] this gives very few frames of data to analyze and in high speed playing, a 30 FPS camera can no longer capture enough information. Higher FPS cameras suffer from lighting issues which significantly impact performance in colour detection.

Motion Blur

One of the main challenges in drum stick tracking occurs because of motion blur. Drumsticks tend the move fast when drumming, the results in frames of video where the position of the stick in unclear. Furthermore motion blur changes the colour of the blurred objects based on their speed and the colour of the background behind them. This occurs as the stick moves while the shutter of the camera is open, resulting in multiple positions of the stick to be captured in a single frame

A key component of drumming that helps combat this challenge is that the sticks tend to pivot around the drummers hand, meaning that the speed comes from rotational movement. This means that the base of the stick tends to move slower than the tip, resulting in less motion blur closer to the base. This means that if the markers at the base can be detected, they can be used to predict the tip of the stick. This method becomes less reliable when the stick as a whole is moving, not just rotating.

Another convenient factor of drumming is that as the stick bounces, or reaches the peak of its movement, it must decelerate and accelerate again. The point in time when the stick moves the slowest (and therefore suffers from the least motion blur) is exactly as a hit occurs. This means a stick is most likely to be detected when a drum is hit.

Rolling Shutter

Rolling shutter is a method of image acquisition used in many digital cameras, where the sensor captures the image sequentially by scanning individual rows of pixels one after another, rather than capturing the entire scene simultaneously. This can result in visual distortions, known as rolling shutter artifacts, when capturing fast-moving objects or during rapid camera motion, as different parts of the image are captured at slightly different times, this means that fast moving drumsticks may not be captured as perfectly straight lines.

Marker Occlusion and Persistance

All markers will not be visible every frame, due to occlusion form the other stick, occlusion from drum kit components or motion blur. When not every marker is visible, some technique must be used to associate visible markers with the actual marker on the stick they correspond to.

Stick prediction

When some markers are occluded, the tip of the stick must still be predicted. A model must be created for predicting the stick tip with limited marker information, irrespective of the distance and angle of the stick to the camera.

2.4 Computer Vision Methods

2.4.1 Camera Calibration

A key component of the intended project is detecting markers on drumsticks and making accurate measurements. However cameras are prone to distortion in the image they capture due to imperfections in the lense shape. Radial distortion is an issue that must be resolved in this project, as it causes straight lines to appear curved. This is an issue as drumstick detection is based around detecting multiple markers in a straight line, therefore removing this distortion is key.

Radial distortion can be represented as follows:

$$xdistorted = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$ydistorted = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Another form of distortion is tangential distortion which occurs when there is imperfect alignment of the lens with the image sensor of the camera, meaning the image plane is not perfectly parallel to the lens plane resulting in a tilted of skewed image. This distortion tends to be less severe than radial but still causes inaccuracies in measurements.

The amount of tangential distortion can be represented as below

$$xdistorted = x + [2p_1xy + p_2(r^2 + 2x^2)]$$
$$ydistorted = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$

To remove the distortion thse following parameters must be found:

$$Distortioncoefficients = (k_1, k_2, p_1, p_2, k_3)$$

To find these parameters we must provide sample images of a well defined pattern (the default being a chessboard). The relative positions of each key point in the pattern are known and their position in the image for each sample image. Using this information and multiple images, the parameters can be solved for and then used to undistort a given image. OpenCV provides functions to calculate these parameters and undistort the image.

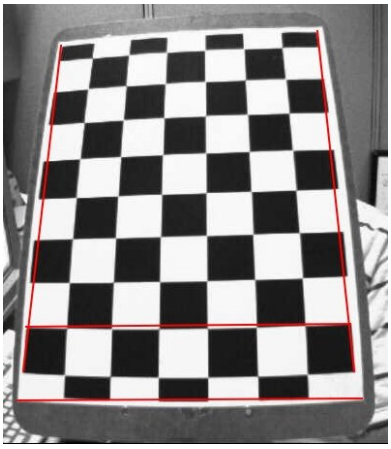


Figure 2.1: Distorted chessboard image

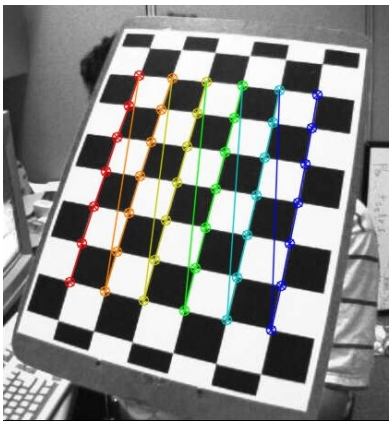


Figure 2.2: Detected chessboard corners

Figure 2.3: Chessboard after undistortion

2.4.2 Background Subtraction

To ensure that misdetection and mistracking of drumsticks is kept to a minimum, only the moving objects in any given frame need to be considered. This means that any pixels that might be misdetected as a drumstick marker can be ignored if they are stationary. Many techniques exists to extract the foreground (e.g moving objects). However the method used in this project is Mixture of Guassians(MOG).[20]

This technique works by modelling the intensity of each channel of each pixel as a mixture of Guassian distributions. A MOG background subtractor is robust against noisy, dynamic, multimodal backgrounds with changes of illumination. This is effective in this particular project as the background will be noisy and dynamic (consider the movement of the cymbals, the drummer and changing lighting conditions) and ideally only the movement of the drumsticks should be considered as foreground.

2.4.3 Morphological Operations

Morphological operations are image processing techniques that process and modify the structure or shape of objects within an image. These operations are particularly useful for removing noise, enhancing object boundaries, and simplifying the overall image structure. The two most common morphological operations are erosion and dilation, often used in combination to perform opening and closing operations [18]. Detecting colored markers in an image can benefit from morphological operations in the following ways:

1. Noise reduction: Images captured by cameras may contain noise, which can interfere with the accurate detection of coloured markers. Morphological operations like opening (erosion followed by dilation) can help eliminate small, unwanted noise elements while preserving the shape and size of the actual markers.

2. Separating overlapping markers: In cases where coloured markers are closely spaced or overlapping, morphological operations like erosion can be used to separate them. Erosion shrinks the objects in the image, creating gaps between adjacent markers, making them easier to detect as individual entities.

3. Filling marker gaps: If coloured markers have gaps or holes within their structure, the closing operation (dilation followed by erosion) can be used to fill these gaps, resulting in a more solid and continuous representation of the marker.

2.4.4 Random Sample Consensus (RANSAC)

RANSAC [15]is an iterative, probabilistic method that aims to find a model that fits a subset of data points with the maximum consensus. It works well when dealing with noisy data containing many outliers. The algorithm randomly samples a minimum set of points required to fit a model, estimates the model, and then determines the number of inliers (points that agree with the model within a predefined threshold). This process is repeated multiple times, and the model with the highest number of inliers is chosen as the final result.

This technique applies well to the drum stick tracking problem. By defining a model based on marker positions, RANSAC can be applied using detected markers as sample points, and determining how well the sampled points fit the model. This technique is robust against outliers, making this algorithm a strong choice for detecting true stick markers, when some markers may be misdetected.

2.4.5 Global Nearest Neighbour

The Global Nearest Neighbour (GNN) algorithm is a data association technique used primarily in multi-object tracking scenarios. The algorithm's goal is to associate measurements from a sensor, such as a camera or radar, with objects being tracked in a scene, considering all possible associations and choosing the most likely one based on a global cost function.

This is key in drum stick tracking, especially marker persistence. When not every marker is detected on a stick (perhaps due to occlusion or motion blur), the markers that have been designated an index, i.e. which marker on the stick the detected marker is. By using a GNN algorithm, the best assignment of detected markers to their position on the stick being tracked can be found. This should allow the stick position to be accurately measured even when all markers are not correctly detected.

Chapter 3

Design

Several choices needed to be made about the design of the system.

3.1 Library choice

3.1.1 Computer vision

OpenCV [6] is the clear choice in image processing library for this project. It efficiently (using C++ bindings) implements common image processing functionality, such as camera calibration, morphological operations, background subtraction and tools for dealing with contours and masks.

3.1.2 Transcription

A few possible music transcription frameworks exist for generating sheet music such as music21,Abjad and Musescore but none of these offer the flexibility of lilypond

3.1.3 Lillypond

LilyPond [4] is the transcription framework for generating sheet music used in this system. LilyPond is a powerful and versatile open-source music engraving software that allows users to create professional-grade sheet music using plain-text input. Lilypond notation can be generated programatically and compiled with Python to a pdf.

LilyPond's notation format for drum music is based on a simple and intuitive text-based syntax. The drum notation is represented using a combination of staff types, note names, and durations, allowing for a clear and concise representation of drum patterns and grooves.

1. **Staff type** To specify that the staff is for drum notation, you need to use the drums command. This command informs LilyPond that the staff should contain percussion notation rather than traditional pitched notes.

2. **Note names** In LilyPond's drum notation, each drum or cymbal is represented by a specific note name. For example, the bass drum is represented by the note name bd, the snare drum by sn, the hi-hat by hh, and so on. These note names correspond to specific positions on the drum staff and are rendered with appropriate symbols or noteheads.

3. **Durations** Note durations in LilyPond are represented by numbers appended to the note names. For example, a quarter note bass drum hit would be written as bd4, while an eighth note snare drum hit would be sn8. Rests can also be included using the letter r followed by the duration, such as r4 for a quarter note rest.

4. **Labelling** The sticking or accent of a note can be added with other specifiers, e.g `sn4->^"L"` , indicates an accented snare quarter note played by the left hand

An example generated lilypond .ly file and its sheet music are shown below.

```
up = \drummode {
<cymr sn >4^"L"^"R" <cymr >4^"R" <cymr >4^"R" <cymr>4^"R"
<hh sn >8^"L"^"R" <hh>8^"L" <hh>8^"L" <hh>8^"L" <hh sn >8^"L"^
   "R" <hh>8^"L" <hh>8^"L" r8
<cymc>1->^"R"
}
\new DrumStaff <<
\new DrumVoice { \voiceOne \up }
>>
```

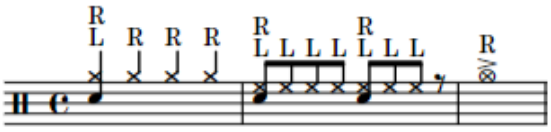


Figure 3.1: Genreated Sheet Music

3.1.4 MIDI Generation

Pygame.midi [7] is the best choice for MIDI generation in this project because it offers a straightforward and easy-to-use interface for creating and manipulating MIDI data in Python. Its extensive documentation and simplicity make it a suitable choice for efficiently integrating MIDI functionality into the project.

3.2 Stick Design

Research was done into the best option for stick tracking for this project. From a frame of video, a drumstick undergoing motion blur must be able to be detected, even if part of it is occluded.

3.2.1 No Markers

Requiring no modification of drum sticks would be an ideal solution, making the system easily accessible to users. This would require an object tracking method that could accurately detect the stick even with motion blur. Whilst object tracking methods have been researched for this kind of situation [17] implementing them and achieving effective results did not seem possible. Furthermore the sticks would need to be initialised as a left stick or right stick if stickings were to be captured and robust object persistence would be needed maintain this data.

3.2.2 Fiducial Markers

Fiducial markers are distinctive and easily recognizable objects, symbols, or patterns used in computer vision. They are widely used as they can express a lot of information; a marker can show 3D pose and encode information (for example a QR code). This would be a very useful tool in drumstick tracking. If multiple markers were used on each stick, occlusion could be accounted for (each marker could express its position along a stick allowing occluded markers position to be predicted), whether the stick is a left or right stick could also be encoded and 3D pose information could be extracted, giving more information to a tracking module and hit detection. However there are major drawbacks to this method. Most notably that the space available for a marker on a drumstick is small, and the sticks will undergo considerable motion blur. While blur resistant fiducial markers have been designed [14], not enough information will be detectable considering the size the markers would have to be and the amount of motion blur. Furthermore printing and attaching numerous fiducial markers to each drumstick is not particularly easy to use by a user.

3.2.3 RetroReflective markers

AeroDrums demonstrate that retroreflective markers can be used for accurate drumstick tip tracking. Low cost methods for object tracking using retroreflective markers exist using phone camera and flash to capture reflection [22]. The low cost method is not robust as it is sensitive to lighting conditions and results are variable depending on the phone camera and flash.

3.2.4 Coloured equally markers

The chosen design is several equally spaced markers placed along the stick. This conveys whether the stick is a left or right stick (by the colour of the markers), allows occluded markers to be predicted (by using position information from the remaining markers), and gives some 3D pose information [13]. The must valuable feature of this stick design is its robustness to motion blur. Drum stick tend to pivot around the base of the stick, resulting in high angular speed at the tip of the stick (the most important part of the stick to track). However, the sticks tend to move slower around the base of the stick, this means that markers at the base of the stick will suffer less from motion blur and can be used to predict the stick tip.

Figure 3.2: Moving and stationary sticks, note the reduced blur at the stick base

3.3 Video format

A few assumptions are made about the video format that must be followed for correct transcription.

1. The sticks are visible in the first frame - This is used to allow the initialise the locations of both sticks
2. The hihat is open in the first frame - This gives a baseline height of the hihat clasp, when the clasp is lower than its initial position, it can be considered as shut.
3. The video is from side on, with the drums on the right - Whilst sticks angled into the camera can be detected and 3D stick acceleration can be estimated, optimal hit detection will occur if the the sticks are parallel with the plane of the image.
4. The first hit played is the first note of the transcription
5. The drummers tempo must be exact - drifting tempo is outside the scope of the project, so it is assumed the drummer will play in perfect time (perhaps to a metronome).

Figure 3.3: Example video setup

3.4 Key Data

The aim of the project is to produce sheet music and a MIDI file representing a video drum performance. To produce that output several intermediate forms of data must be produced. The data flow diagram is shown in figure 3.4. The required data will be briefly described in the following sections.

MIDI File and Sheet Music PDF

These files are generated by the MIDI generator function and transcriber function described in this chapter, it requires the **framesHitDict** list as an input to dedscribe when and how hits occured.

FramesHitDict

The FramesHitDict data structure is a dictionary that uses a frame number as a key to access an array of drum hit information for that frame so that:

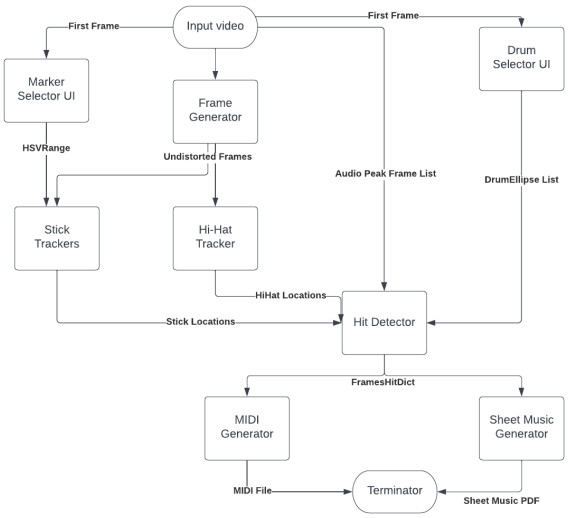


Figure 3.4: Data Flow diagram

```
framesHitDict[frameNum] = [hitNotationDict]
```

It is used by the result generation functions to get the hit drum information for each frame where a hit occurred.

hitNotationDict

This is a data structure that records the drum hit information in the following dictionary:

```
{'notation': String,
'subdiv':int,
'sticking': String,
'accent': Bool,
'midiPitch':int}
```

1. **notation** - The lilypond notation used to represent that drum to generate sheet music. For example 'sn' for a snare hit, 'hh' for a hihat hit.

2. **subdiv** - A integer value representing the subdivision value of the note. This is initialised to the smallest subdivision used (16 to allow for semiquavers). This value is manipulated by the result generation functions to calculate the notes value for correct transcription.

3. **sticking** - Either 'L', 'R' or '' depending on the stick that was used to hit the drum or whether sticking is being used.

4. **accent** - A boolean value representing whether a accent should be included in the sheet music for that hit

5. **midiPitch** - An integer value used by the MIDI standard to define the drum sound to be played.

The hitNotationDict values are found by the HitDetector object and used to populate the framesHitDict for used by the result generation functions.

DrumEllipse List

This is a list of drumEllipse types. It represents all the drums in the frame. It is used by the HitDetector object do calculate which drums have been hit. Its values are:

1. **notation**
2. **midiPitch**
3. **ellipse** - an OpenCV2 ellispe object defined by center_coordinates, axesLength, angle, startAngle, endAngle

This list is populated by a simple UI that allows users to define an ellipse representing each drum.

Stick and Hihat Location Data

The acceleration and location data for each marker on a stick is calculated by the StickTracker objects. The location of the hihat clasp is found by the hihat tracker object. The stick location data is used by the HitDetector object to detect drum hits and the hihat clasp location is used by the HitDetector to determine whether a hihat is open or shut.

Audio Peak Frames

This is a list of frames where peaks in audio amplitude accuaretly found from analysing the video audio. It is optionally used by the hitDetctor object to reject hits that do not occur within a few frames of an audio peak.

Undistorted / foreground frames

To extract stick and hihat location from the video, undistorted frames, in the HSV colour space, with the backgorund removed must be passed to the respective tracker objects. The FrameGenerator object is used to generate theese frames to be passed to the trackers.

How this key data is found is describe in the next chapter.

Chapter 4

Implementation

4.1 Frame Generator

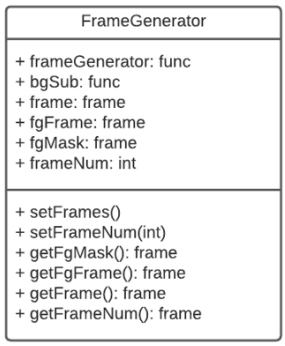


Figure 4.1: Frame Generator Object

The frame generator object is used to retrieve the next undistorted HSV frame in the video. It also provides these frames to a mixture of gaussian's background subtraction object. This allows both the undistorted frame and the undistorted frame with the background removed to be fetched from this object.

Parameters

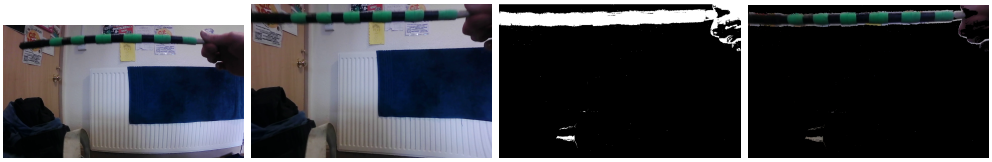
frameGenerator - This parameter is a function that can be called to output the next frame. It is initialised with the path to the video being processed and the path to a file containing the camera calibration coefficients. By using the distortion coefficients a mapping is found for every pixel to find its correct position in the frame. This requires doing some interpolation as the mapping may give non-integer pixel locations. Finally the image is cropped so that the final image is rectangular. A function is generated that applies this mapping and cropping to the next frame of video every time it is called. The code to generate this function is shown in 6.3
bgSub - This parameter is an OpenCV mixture of Guassians background subtracter object. When supplied with a frame of video, it can return the foreground mask. So that the foreground can be extracted from the fist few frames, the background subtracter object us initialised with the first few frames of video to learn the representations of background pixels.

frame,fgrame,fgMask - These parameters tore the undistorted frame, the the foreground frame and the foreground mask respectively.

Methods

setFrames() - This function is called to update all of the parameters. It sets the next frame, updates the bgSub object and updates the frame values.

Getters - The remaining methods can be called to access the current undistorted frames store in the FrameGenerator object.



(a) Distorted Frame (b) Undistorted framework (c) Foreground Mask (d) Foreground Frame

4.2 Stick Tracker

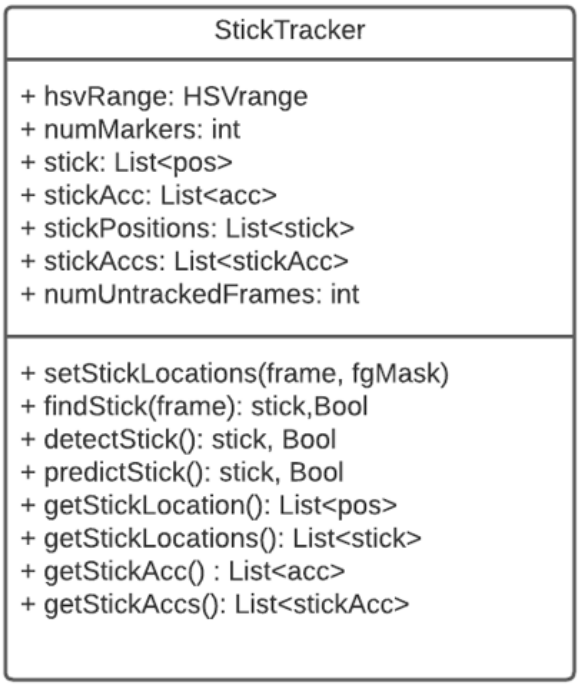


Figure 4.2: Stick Tracker Object

The stick tracker is initialised with the colour range defining the markers to be tracked. It is passed undistorted foreground frames and calculates the most likely stick position for each

frame and its associated acceleration. This data is passed to the hit detector object to determine if a hit has occurred.

4.2.1 Parameters

1. **hsvRange** - This parameter is a HSVRange type, two HSV pixel values, a lower and upper bound. Pixel values between the two bounds are detected as potentially being part of a stick marker. These values are found by a simple UI. The user can select the regions in the image that correspond to stick markers. The mean and standard deviation colour of the selected regions is calculated and the HSVRange is defined as 2 standard deviations from the mean colour. Only the hue and saturation channels are considered as large variation in the brightness channel is acceptable as this accounts for changing brightness due to lighting and motion blur, the value channel range is fixed between 50 and 200. The floodfill algorithm is used to select a region of similar colour around where the user clicked.

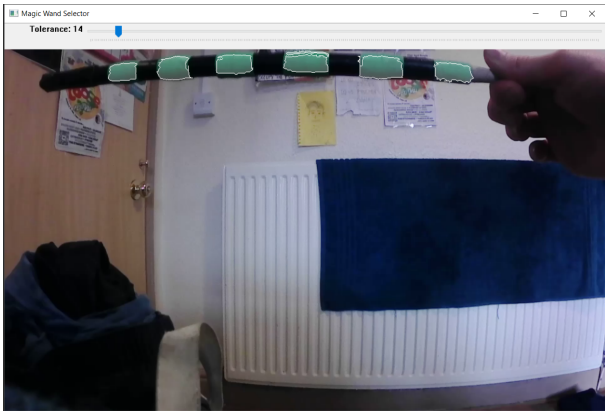


Figure 4.3: Colour Selection UI

2. **numMarkers** - An integer value defining the number of markers on a stick. This is fixed to 6 for use in this report.
3. **stick** - A list of position values that represent the positions of each marker on the stick for the current frame, initialised to None.
4. **stickAcc** - A list of acceleration values for each marker.
5. **stickPositions** - A list of stick values for every frame. Stores all previous stick locations.
6. **stickAccs** - A list of all previous stick acceleration values.
7. **numUntrackedFrames** - An integer recording the number of frames the stick has not been found in. This is used to predict stick positions for undetected frames.

4.2.2 Methods

setStickLocations(frame,fgMask)

This method is called every frame to update the stick locations.

```
def setStickLocations(self,frame,fgmask):
    bgSub = bitwise_and(frame,fgmask)
    markerContours = findContours(bgSub,self.HSVRange)

    stick,found = self.detectStick(markerContours)
    if not found:
        stick,found = self.predictStick(markerContours)

    if not found:
        self.numUntrackedFrames += 1

    if found:
        self.tweenStick()
        self.stick = stick
        self.stickPositions.append(stick)
```

It attempts to find the stick by detection and tracking and records the position (as well as tweening missed frames) if successful. If the stick could not be located for tracking, the number of untracked frames is incremented so that its locations can be predicted by tweening when the stick is next detected.

Colour detection

Candidate markers are found by finding contours of the defined hsvrange, rejecting contours below a certain area (accounting them as noise) and applying opening and closing morpholgical operations to ”clean” the detected markers. The final contours returned for processing are the convex hull of each marker found. This is useful for a more accurate estimation of the actual marker shape. The contour centres are used for stick detection, so the convex hull gives a better approximation of the true centre of the marker. The detected marker contours are shown in figure 4.2.2

Figure 4.4: Detected, filtered coloured markers

Detect Stick

In object tracking problems the object must be initially detected so that it can be tracked. The tracking algorithm uses its previous location to inform its prediction of its next location.

Because of this initial detection must be robust and accurate.

A simple RANSAC algorithm was created to detect marker positions described below:

1. Define a line by randomly selecting two contour centres
2. Calculate the distance of every contour centre to the line
3. Create a list of inliers - contour centres that are below a threshold distance from the line
4. Calculate inliers score and store the inliers with the lowest score
5. Repeat until the maximum number of iterations has been exceeded
6. Return the inliers with the minimum score if it is below inf, else no stick has been detected

The inlier scoring function must reject any set of inliers that do not represent a stick. The scoring function is defined by :

$$Score = NumScore(points) + ModelScore(points) + DistScore(points)$$

This score is based on three considerations:

1. Number of inlier points (Must be at least equal to the number of markers on the stick)

$$NumScore(points) = \left\{ \begin{array}{ll} inf, & \text{for len(points)} < \text{numPoints} \\ 0, & \text{else} \end{array} \right\}$$

2. How well the potential markers fit a a stick model

$$ModelScore(points) = \left\{ \begin{array}{ll} inf, & \text{for stickModelError(points)} > 1 \\ 0, & \text{else} \end{array} \right\}$$

3. Distance of each point to the line

$$\sum_{points} (dist(point, line))$$

This scoring function produces a score below inf if there are enough inliers and they are in an acceptable arrangement to represent a stick. Once an acceptable stick has been found, further iterations aim to minimise the distance of the markers to the line, increasing its accuracy.

Track Stick

If the strict requirements for a stick to be detected are not met, the system will attempt to predict a stick position from the available information. This is done by transforming the previously found stick to match the new information. This requires two markers to be detected in the new frame and to find which marker they correspond to from the previously detected stick.

GNN marker assignment

A greedy global nearest neighbour algorithm is used to associate two markers with with markers from the previous frame. The distance matrix between every detected contour and marker in the previous frame is found. An assignment of contours to marker indices is found such

that the stick predicted from the detected markers is a minimal distance from the previously detected stick.

Stick Translation

The goal of this section is to determine the transformation that best aligns the previously detected stick with the new contours. The general 2D rotation matrix **R** and translation matrix **T** can be represented as follows:

A 2D rotation matrix **R** for an angle θ is given by:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

A 2D translation matrix **T** for a translation vector $\mathbf{t} = (t_x, t_y)$ is given by:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

To apply both rotation and translation to a 2D point $\mathbf{p} = (x, y)$, we first augment the point with a 1, forming a homogeneous coordinate:

$$\mathbf{p}' = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Now, we can create a combined transformation matrix **A**:

$$\mathbf{A} = \mathbf{T}\mathbf{R}'$$

Where $\mathbf{R}'$ is an augmented version of the rotation matrix:

$$\mathbf{R}' = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Finally, we apply the combined transformation to the augmented point $\mathbf{p}'$:

$$\mathbf{p}'' = \mathbf{A}\mathbf{p}'$$

The transformed point $\mathbf{p}'' = (x'', y'', 1)$ can then be obtained by de-augmenting $\mathbf{p}''$.

By applying the translation and rotation operations to the previous stick position array, the new tracked stick position can be estimated

Tween stick

When a stick position can not be detected or tracked it is marked to be tweened. The next time the stick is detected, the undetected stick poitions are calculated by linear interpolation.

$$stepSize = (NewStick - OldStick)/numUntrackedFrames$$

$$stick_n = OldStick + stepSize * n$$

4.3 Hi-Hat Clasp Tracker

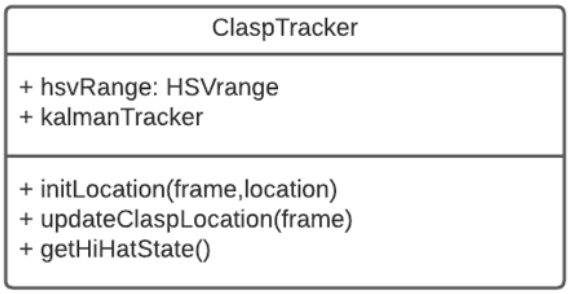


Figure 4.5: Clasp Tracker Object

The hihat clasp is tracked by a small coloured marker. It is assumed that in the first frame of video the hihat will be open. The clasp tracker object tracks the clasp marker and if it is threshold value below its starting position then it is considered shut, otherwise it is considered open.

Kalman filter

As sticks will be moving past the hihat clasp, occlusion is inevitable. By simply tracking the clasp marker contour, occlusion will modify the detected hihat position, and so its state. To account for occlusion, the clasp location is modified by a kalman filter which is passed the markers contour centre each frame. By using a kalman filter, a single frame of occlusion, modifying the markers centre point wrongly will not immedietly effect the clasp position. However, if the hihat position stays changed over a few consecutive frames, the kalman filter will move its recorded position towards its new state.

4.4 Hit Detector

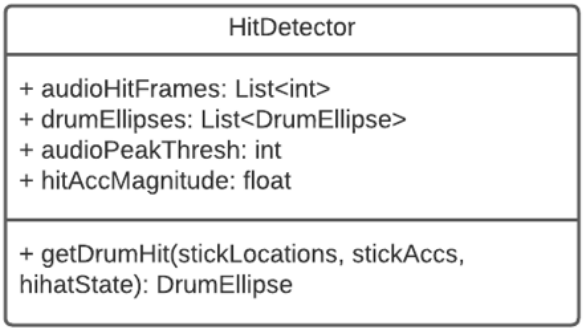


Figure 4.6: Stick Tracker Object

The hit detector object analyses the location data of each stick on each frame to determine if a drum has been hit, it also considers the hihat state to determine if a hihat hit is an open or

closed hihat hit. Each frame, it returns a list of drums hit that frame, along with any metadata to be recorded in the transcription (e.g. which stick was used, whether the hit was an accent).

4.4.1 Drum selection

The hit detector object is initialised with the the user defined drum information for a given video. Creating a hit detector object prompts the user to define the drums present in the video, and the region they occupy. This is done by a simple UI where the user selects points on the edge of each drum and the ellipse that bounds that area is recorded. The information that defines each drum is saved as a **drumEllipse** object. This object contains the ellipse that defines the drum region, the lilypond notation used for generating the sheet music using that drum and the MIDI pitch value used for generating the MIDI file. The hit detector object then stores a list of all drumEllipse objects in the scene.

(a) Ellipse selctor (b) Drum Selector UI

4.4.2 Audio peak detection

This optional feature is used to reject misdetected hits. If being used, the hit detctor object will be initialised with the audio of the video to be anlysed and will detect audio peaks in the file, find the frames where the volume peaks, indicating a hit has taken place. A list of all audio peak frames is stored as property of the object. If a hit is visually detected, the hit detector will check that the hit occured close to an audio peak and only record the hit if it is.

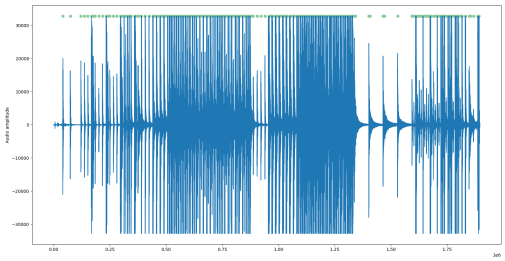


Figure 4.7: Audio amplitude against time with audio peaks marked

The inclusion of audio techniques is discussed in the conclusion. Simple amplitude peaks are used for onset detection verification but more nuanced methods exits that would provide better results (for example, amplitude peaks would not work if other music was being played during the performance)

4.4.3 Hit Detection

The actual detection of hits works by considering acceleration of the stick tip for each frame. Points where the magnitude of accelration is high, implies a sudden change in direction. This change in direction can be a result of a fast moving drumstick colliding with a drum, or the drummer very quickly "whipping" the stick in a new direction (this occurs at the peak of the sticks height, when the drummer brings the stick down).

Several requirements are needed to classify a stick movement as a drum hit:

1. The tip location is close to or inside the drum ellipse
2. The magnitude of acceleration in the direction perpendicular to the major axis of the drum ellipse of above a threshold
3. The tip was above the drum in the previous frame
4. The hit occurred within a window of an audio peak (if audio peaks are being used).

Each requirement is checked each frame for each drum and if they are all met, a drum hit and its metadata is recorded. The list of drum hit information is then passed to the result generation modules.

4.5 Result Generation

4.5.1 Sheet Music Generation

Creating a lilypond drum sheet music requires writing a lilypond ".ly" file using the lilypond sheet music description format. The key information that needs to be calculated is the note value for each drum being recorded, where rests are needed and the value of each rest.

Frame to Beat Conversion

The frameHitDict passed to the transcription function must be converted to a beatHitDict where the beat number for each hit is calculated. It is assumed that the first drum hit will be the first beat.

$$beatNum = (60 * beatsInBar * FPS)/(BPM * subdivion * (frameNum - minFrameNum))$$

Note value

A recursive algorithm is used to calculate the beat and rest values for each beat in each bar. The calculation can be summarised by - **"Replace a note followed by a rest of the same value, with the first note with half the value."**

An example 8th note bar is resolved below:

$$[sn8, r8, r8, r8, sn8, r8, sn8, r8]$$

$$[sn8, r8, r8, r8], [sn8, r8, sn8, r8]$$

$$[sn8, r8], [r8, r8], [sn8, r8], [sn8, r8]$$

$$[sn4], [r4], [sn4], [sn4]$$

$$[sn4, r4], [sn4, sn4]$$

$$[sn2], [sn4, sn4]$$

$$[sn2, sn4, sn4]$$

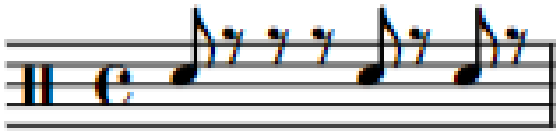


Figure 4.8: Unprocessed snare bar

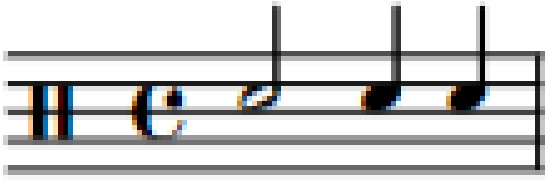


Figure 4.9: Processed snare bar

The code to calculate these note values is shown in 6.3

Simple string manipulation is then used to convert the data associated with each hit and rest to the lilypond format, apply accents and stickings, and then compile the file to generate a pdf.

4.5.2 MIDI Generation

MIDI generation is a relatively simple task. The tempo of the file (which is user supplied) must be set, as well as the instrument being used, a drum kit. The beat number of each hit has been calculated by the transcription phase. After beat numbers for each hit have been calculated, each hit is added to the track using pygame MIDI functionality. The final file is then written to the disk and can be used by the user. This code is shown in 6.3

Chapter 5

Evaluation

5.1 Key Results

Shown in figures 5.1,5.2,5.3, 5.4 are some results generated using the process described in this report. These results capture variation in timing, what drum was hit, whether the hihat was open, and the stickings of the groove. Furthermore for each item of drum notation an equivalent MIDI file is also created. By eye, these transcriptions are mostly accurate and represent each groove accurately. This shows that the system in capable of producing accurate transcriptions of a drum performance with marked sticks from a side on view.

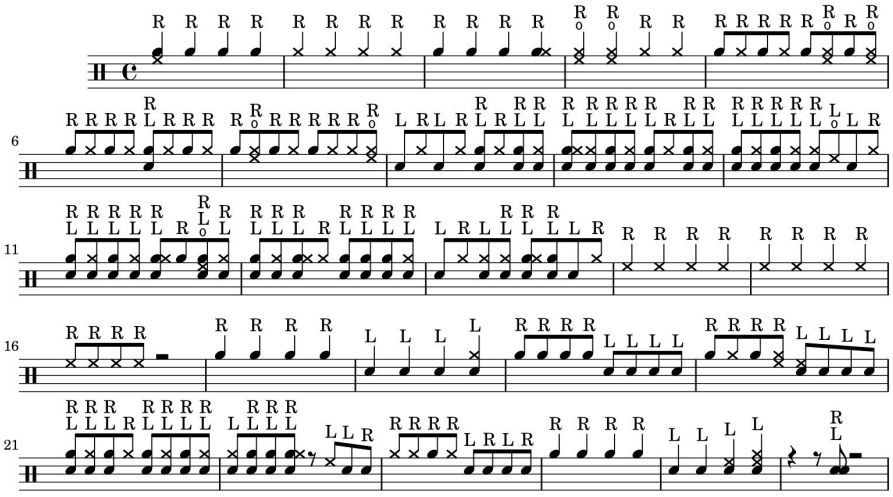


Figure 5.1: Generated sheet music

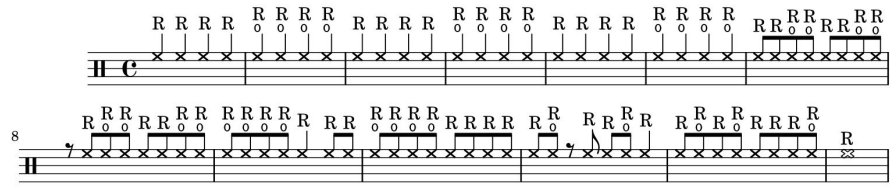


Figure 5.2: Generated sheet music

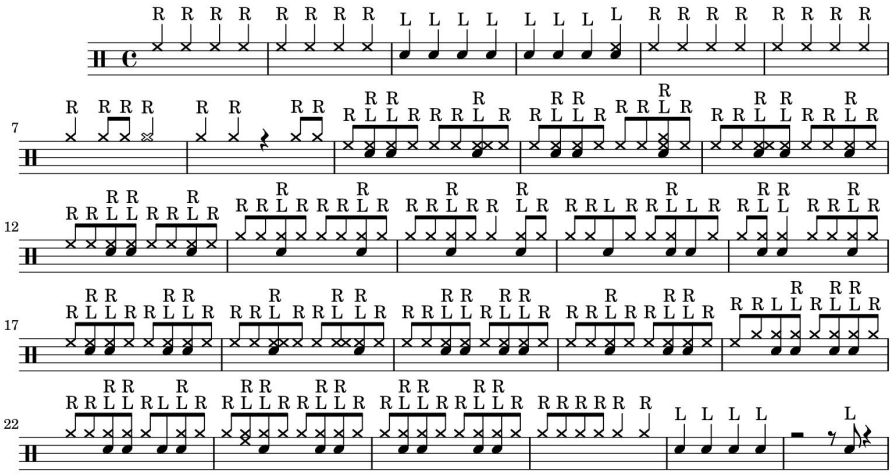


Figure 5.3: Generated sheet music

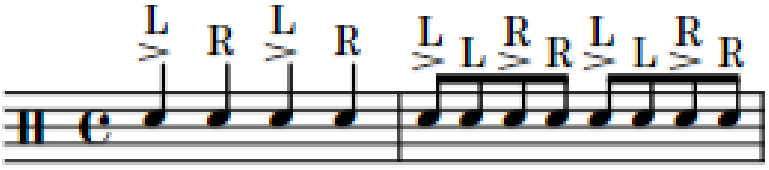


Figure 5.4: Generated sheet music

For each of the aims defined in section 1.3 an experiment will be performed to measure the performance of this system in that respect

5.2 Track two fast-moving blurry marked drumsticks

1. Detect a stationary drum stick marked with coloured markers
2. Track a fast moving stick that suffers from motion blur

5.2.1 Detect a stationary drum stick marked with coloured markers

Detecting a stick is a key part of the system. A stick must be initially detected for tracking to begin, and if misdetection occurs in the tracking module, detection is used to correct tracking (reinitialising the stick to its correct positon). The detection module is built to detect a stick when it is stationary and not suffering from motion blur, the following tests will measure the effectiveness of the stick detection module.

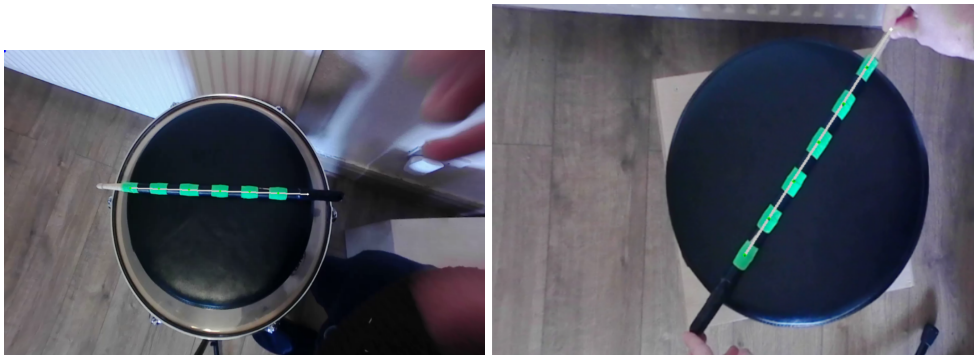
Stick distance

The first factor considered in detection was distance from the camera. The camera was slowly moved away from a stationary stick and the distance at which the stick detection failed was measured. This occurred when a single marker took up less than 1% of the width of the image. This is an issue with the noise rejection implemented. As any contours under a certain size are rejected, if the stick is too far away from the camera, its markers will be rejected as noise.

Stick angle

The two relevant rotations of the stick were considered (rotating in the plane of the camera, and toward the plane of the camera).

As expected (and shown in 5.2.1) the stick tip is only correctly detected in an 180 degree range as the tip of the stick is hard-coded to be predicted to the right of all markers. The stick detection can be improved to correctly detect sticks with the tip on the left or right side, this is especially important as left handed drummers may have a different kit layout and as such need to film a video with the drumstick tip on the left side.



(a) Incorrect tip detection (b) Correct tip detection

Figure 5.5: Rotated stick detection

5.2.2 Track a fast moving stick that suffers from motion blur

When the stick is moving it suffers from motion blur and rolling shutter. Some markers may not be detected, adn the detected markers may not fall exactly on a straight line. The tracking module deals with predicting the stick position from this limited information.

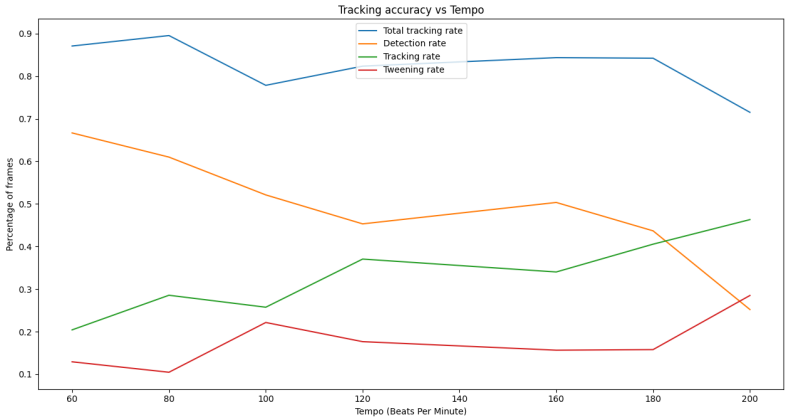


Figure 5.6: Tracking accuracy VS Tempo

Figure 5.6 shows the percentage of frames that are detected, tracked and tweened as the tempo increases. It is clear that as tempo increases, the number of frames the stick can be detected in decreases significantly. This is to be expected as a limiting factor of visual based tracking is the frame rate, and the amount of motion blur. However the number of tracked frames stays quite high even at high tempos.

5.3 Detect when a specific drum is hit and how

1. Detect whether the hi-hat is open or closed
2. Detect whether a hit is an accent

5.3.1 Detect whether the hi-hat is open or closed

The hihat state accuracy is described by the percentage of frames in which the hihat state is correctly determined. This is done by hand defining the ranges of time in which the hihat is open or closed for a test video. The measured hihat state accuracy over 5 videos is shown in table 5.3.1. On average the correct hihat state is detected 0.64% of the time. This may be due to the Kalman filter resulting in the predicted hihat state lagging behind the actual state and hihat clasp occlusion that is too drastic to be combatted by the Kalman filter.

		Predicted	
		Open	**Closed**
Actual	**Open**	True Open (0.68)	False Closed (0.32)
	Closed	False Open (0.39)	True Closed (0.61)

5.3.2 Detect whether a hit is an accent

The accent accuracy was measured over 100 snare hits at 60 bpm, where 50% of hits were accented. The confusion matrix shows strong accuracy for detecting snare accents. It is worth

noting that accent detection is sensitive to stick distance and angle so accent detection accuracy will be lower in less controlled experiments.

		Predicted	
		Accented	**Unaccented**
Actual		True Accented (0.82)	False Unaccented (0.1)
		False Accented (0.18)	True Unaccented (0.9)

5.4 Generate MIDI and drum notation representations

1. Produce correct sheet music that records the groove played
2. Produce a MIDI file that corresponds to the groove played

A manual transcription will be done for each groove being tested. The percentage of correctly transcribed notes will be recorded, and number of misdetected notes will be recorded. As the MIDI file is based on the same data as the sheet music, this metric will evaluate both subgoals.

Figure	Notes Transcribed	Notes Played	Correctly Transcribed Notes	Misdetected Notes	Transcription Accuracy	Misdetection Rate
5.1	232	210	192	30	0.91	0.13
5.2	69	72	65	0	0.94	0
5.3	222	218	211	11	0.95	0.049

Table 5.1: Transcription Accuracy and misdetection rate

The table shows that in the examples included, above 90% of drum hits are correctly transcribed. However misdetection of hits is an issue, in groove 5.1, 13% of all transcribed notes were wrongly detected and should not have been included.

Chapter 6

Conclusion

The results show clearly that using the methodology described in this report, accurate MIDI and drum notation can be produced from a side on video of marked drumsticks. This method provides advantages over traditional audio based methods in that it can more readily capture nuance in playing. In this implementation it is shown that accents and stickings can be captured. While this report certainly demonstrates the potential of visual based drum transcription, the current method has significant drawbacks that can be improved upon to make a more robust system.

A list of the most limiting drawbacks are described below and this chapter will describe possible areas of research, and technologies that could be used to overcome them to produce an accessible, robust and nuanced drum transcription system.

1. **Drum stick speed** - As demonstrated in the evaluation section, the FPS of the camera used is a key limiting factor in capturing a drum performance. For a drummer playing single notes at 150BPM the tip of the drumstick can easily reach 15 m/s [9] with higher speeds being very common. The current system will not record a drum hit unless a frame is captured where the tip of the stick is detected to be inside of a drum region. Even by indirectly tracking the faster moving stick tip by tracking the slower moving base of the stick, a method of predicting stick position in between frames and deducing whether a hit has occurred or using a higher FPS camera would be necessary for high speed performance capture.

2. **Camera angle** - The described methods requires the camera to view the performance from side on. This is because it only considers 2D acceleration to detect a hit and because the model used to describe a stickis not accurate at tracking a stick that is angled into the camera, it works best when markers are equally spaced and not distorted by perspective. This fixed camera angle limits the functionality of the system as it is difficult to include a full drum kit into a side on video without some key parts of the kit being occluded. The equally spaced markers give perspective information, the spacing detected by a camera informs the angle of the stick into the camera. A method that leverages this information (or any other methods that can consider the 3D position of sticks) would allow for dynamic positioning of the camera to capture more information.

3. **Motion blur/rolling shutter** - A separate issue from stick speed resulting in frames not capturing key information, is the issue of detecting markers when motion blur occurs. The colour and shape of the markers is distorted. The current method deals with these

issues by taking advantage of the less distorted markers at the base of the stick and using a wide colour range to still detect colour distorted markers. This method has its limits as too large a colour range cannot be used as misdetecting too many other regions of the image as markers pixels would affect accuracy, and too high a speed will distort the markers too much for the current method to detect them. For a more robust system that can deal with high speed drumming, a more nuanced method would need to be used to deal with this issue.

4. **Bass drum capture** - A key part of a drum kit is the bass drum, a drum played by pressing a pedal with your foot. This drum is heavily used in most performances and unfortunately tends to be occluded from view. Because of this a visual method cannot capture bass drum hits. Because of this any system that intends to capture bass drum hits must use some audio based techniques to capture these hits and produce a full transcription of the performance.

5. **Non probalistic model** - The state of the art in object tracking methods is to use probabilistic models, e.g. assigning a probability to all possible hypothesis and choosing the best one, this allows multiple possibilities of object position to be considered, and to be updated with new information. The method described in this report is deterministic and only considers 1 possibility for position at each frame, this propagates any errors in tracking forward which would only be resolved by a correct detection step.

6.1 Areas Investigated

6.1.1 3D Stick Positioning

By being able to track a sticks positing in 3D, the issue of variable camera angles could be resolved. understanding the angle and position of the stick would allow a stick to be detected from a much larger range of viewpoints, and its 3D acceleration data could be used to detect hits from a non-side on angle. The initial motivation behind the use of equally spaced coloured markers as a tracking aid was twofold, to combat motion blur at the tip and to provide 3D positioning information. The idea behind this is the measured distance between each marker as captured by the camera would give the angle of the stick into the camera and the measured length of the stick would give the distance of the stick from the camera. Using this information a representation of the 3D position of the stick could be found and that information could be used to detect drum hits in 3D. While this hypothesis is correct and supported by literature [13] I found that the level of accuracy in markers position detection needed for sensible 3D results is much too high for use in high speed tracking.

Pose estimation is a well researched area in computer vision, and by detecting the corners of a 2d object (e.g QR code), its 3D pose can be accurately determined with 6 DOF from a single camera using a technique called PnP-RANSAC [10]. However the only requirement for this method is ”at least 3 non-collinear points” meaning 3 points that do not lie on the same line, which does not work for the collinear stick markers. A common method for capturing 3D information from fixed camera is by using stereo cameras [23]. While this method would accurately provide useful 3D information it requires calibration to set up and two cameras. Using this method would not be as accessible as a single camera setup. Because of this a stereo camera method was rejected.

In hindsight the exact 3D position of each stick is not necessarily important and only the 3D angle of the stick would need to be known, not its 3D position. 3D acceleration data could be used to detect a hit and potentially its 2D position could be used to determine the drum that was hit. Further research should be done into extracting 3D information from video to improve ths system as a whole.

6.1.2 Simple audio methods

While the aim of this project was to investigate visual methods for drum transcription, it is clear that a combination of visual and audio methods would be beneficial since most video cameras tend to capture audio as well. The most beneficial result of using audio (apart from the previously discussed audio peak detection as an aid for onset drum onset detection) would be to include bass drum hits in transcriptions.

A small investigation into a band pass filter method [12] was conducted. The aim being to find a characteristic frequency band that defined a bass drum and no other drums. A small training set was created using audio recordings of each drum on a kit and using a differential evolution method to learn a combination of filters that accurately captured bass drum hits. While the method resulted in live detection of bass drum hits for a low processing cost, is incorrectly detected bass drum hits too often to be reliable. Furthermore it involved creating a custom training dataset for a specific drum kit. This method was eventually rejected as the state of the art in audio transcription tends to use other methods.

Similarity measures between audio samples was briefly investigated by comparing the fourier transform of a sample audio recording and a template bass drum hit. This method was also found to be unsuccessful and bass drum detection was determined to be out of scope for the project.

It is clear that audio transcription methods for drum are successful [21] and further work into combining both visual and audio methods for complete transcription is a clear next step for the project.

6.1.3 Motion blur research

6.2 Future considerations

A number of quality of life changes could be made to the system as is. These changes would not be as impactful as including 3D information, or combining audio methods but would still improve, usability, accuracy and robustness.

NLP based groove analysis - Using NLP methods and considering drum grooves as a ”language” methods (e.g. LSTMs,RNNs etc) can be used for predicting the most sensible drum hits in a groove [19]. Combining an output probability of a drum being hit based on the context of the groove so far (and training data of common groves) and a probability of a stick hitting a drum based on visual methods could be used to reject misdetected hits that make no sense in the current context.

Auto drum detection - In the current implementation the drum ellipses that the system checks for a hit within are user defined. However curve detection is a relatively trivial problem in computer vision and drum detection could be achieved by methods such as the generalised Hough transform [8] using edge detection. As the general layout of a drum kit is standard, the labelling of each detected drum could potentially also be automated. Another method

for automatic drum detection is described in [11]. By using audio transcription, the regions in which drumsticks lie when an audio hit is discovered can be used to determine the drum regions. This data could then be used to iteratively improve the audio detection using visual detection methods.

Auto stick calibration - The current implementation requires the user to manually select the regions of colour that correspond the marker colours. Methods such as template matching could be sued to detect a stick in an image frame and find the colour of the markers to track. This method would reduce the input needed for the user, making the entire transcription process easier to use.

6.3 Summary

To conclude, a methodology for drum transcription has been created, that allows a user to track drumsticks, detect drum hits and transcribe the performance to two standardised formats, MIDI for digital use, and drum notation for sharing the groove with other drummers who read sheet music. The results show that some nuance in performance can be captured, stickings and accents, that cannot by the the generally used audio methods. With further work into considering 3D information about drumsticks, dealing with motion blur and combining the novel visual-based methods describe in this report with the state of the art audio methods available, a robust, accessible single camera transcription system is clearly possible.

Bibliography

[1] Aerodrums. https://aerodrums.com. Accessed: 2023-04-16.

[2] Drumstik. https://drumstik.com/. Accessed: 2023-04-16.

[3] Freedrum. https://freedrum.rocks/. Accessed: 2023-04-16.

[4] Lilypond. http://lilypond.org/. Accessed: 2023-04-16.

[5] Midi specifications. https://www.midi.org/specifications. Accessed: 2023-04-16.

[6] Opencv documentation. https://docs.opencv.org/4.x/. Accessed: 2023-04-16.

[7] Pygame.midi documentation. https://www.pygame.org/docs/ref/midi.html. Accessed: 2023-04-16.

[8] D. Ballard. Generalizing the hough transform to detect arbitrary shapes, 1981.

[9] S. Dahl. Measurements of the motion of the hand and drumstick in a drumming sequence with interleaved accented strokes - a pilot study.

[10] H. U. Eric Marchand and F. Spindler. Pose estimation for augmented reality: a hands-on survey.

[11] O. Gillet and G. Richard. Automatic transcription of drum sequences using audiovisual features. In *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 3, pages iii/205–iii/208 Vol. 3, 2005.

[12] A. Maximos, A. Floros, M. N. Vrahatis, and N. Kanellopoulos. Real-time drums transcription with characteristic bandpass filtering. In *Proceedings of the 7th Audio Mostly Conference: A Conference on Interaction with Sound*, AM '12, page 152–159, New York, NY, USA, 2012. Association for Computing Machinery.

[13] M. Najafi and R. Rohling. Single camera closed-form real-time needle trajectory tracking for ultrasound.

[14] M. G. Prasad, S. Chandran, and M. S. Brown. A motion blur resilient fiducial for quadcopter imaging. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 254–261, 2015.

[15] R. Raguram, J.-M. Frahm, and M. Pollefeys. A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In D. Forsyth, P. Torr, and A. Zisserman, editors, *Computer Vision – ECCV 2008*, pages 500–513, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[16] P. Roy, F. Pachet, and S. Krakowski. Improving the classification of percussive sounds with analytical features: A case study. pages 229–232, 01 2007.

[17] C. Seibold, A. Hilsmann, and P. Eisert. Model-based motion blur estimation for the improvement of motion tracking. *Computer Vision and Image Understanding*, 160:45–56, 2017.

[18] P. Soille. *Opening and Closing*, pages 105–137. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[19] C. Southall, R. Stables, and J. Hockman. Automatic drum transcription using bi-directional recurrent neural networks. In *International Society for Music Information Retrieval Conference*, 2016.

[20] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking.

[21] C.-W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller, and A. Lerch. A review of automatic drum transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(9):1457–1483, 2018.

[22] D. Yamaji, H. Hakoda, W. Yamada, and H. Manabe. A low-cost tracking technique using retro-reflective marker for smartphone based hmd. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, page 1–6, New York, NY, USA, 2018. Association for Computing Machinery.

[23] T. Y. . F. T. Yasushi Sumi, Yoshihiro Kawai. 3d object recognition in cluttered environments by segment-based stereo vision.

FrameGenerator code

```
def getFrameGenerator(videoPath,calibrationPath):
    #Returns a funciton that can be called to get the next
        frame of the undistored video given by the path

    CAM,DIST = cameraCalibration.load_coefficients(
        calibrationPath)
    cap = cv2.VideoCapture(videoPath)
    w,h = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)),int(cap.
        get(cv2.CAP_PROP_FRAME_HEIGHT))
    newcameramtx, roi=cv2.getOptimalNewCameraMatrix(CAM,
        DIST,(w,h),1,(w,h))
    mapx,mapy = cv2.initUndistortRectifyMap(CAM,DIST,None,
        newcameramtx,(w,h),cv2.CV_32FC1)

    def frameGenerator(frameNum=False):
        if frameNum:
            cap.set(1,frameNum)
        ret, frame = cap.read()
        if frame is not None:
            frame =cv2.remap(frame,mapx,mapy,cv2.
                INTER_LINEAR)
            x,y,w,h = roi
            frame = frame[y:y+h, x:x+w]
            return frame
        return None

    return frameGenerator
```

Note Value Calculation Code

```
def simplifyBar(drumArray,depth):
    #Input: drumArray - a list of length subdivision that
        contains either a drum notation or a rest
    #Uses combining rules to convert the list to a list of
        lillypad, notelength drum notatoins
    #e.g [r,r,sn,r] -> [r2 sn4 r 4]
    #(ToDo - length 1/2 notes look ugly, perhaps length 4 is
        minium, use edge case)
    if len(drumArray) == 2:
        a = {'notation':drumArray[0],
            'subdiv':2**depth}

        b = {'notation':drumArray[1],
            'subdiv':2**depth}
```

```
        return combineNotations([a],[b])

    drumArrayLength = len(drumArray)
    firstHalf, secondHalf = drumArray[0:int(drumArrayLength/2)
        ], drumArray[int(drumArrayLength/2):]

    return combineNotations(simplifyBar(firstHalf,depth+1),
        simplifyBar(secondHalf,depth+1))

def combineNotations(a,b):
    #Takes two notation dicts (with keys notation and subdiv)
        and combines them to a reduced subdiv notation

    #If the length of a or b is more than one, it cannot be
        reduced further
    if len(a) != 1 or len(b) != 1:
        return a + b

    if b[0]['notation'][0]['notation'] == 'r':
        return [{'notation':a[0]['notation'],
            'subdiv':int(a[0]['subdiv']/2)}]
    else:
        return a+b
```

MIDI Generation

```
def createMidi(subDivDict,tempo):
    smallestSubDiv = (min(k for k, v in subDivDict.items()))

    # create your MIDI object
    mf = MIDIFile(1)     # only 1 track
    track = 0   # the only track

    time = 0    # start at the beginning
    mf.addTrackName(track, time, "Sample␣Track")
    mf.addTempo(track, time, tempo)

    # add some notes
    channel = 9
    volume = 100

    for beatNumber in subDivDict:
        time = beatNumber - smallestSubDiv    # start on beat
            0
        duration = 1      # 1 beat long

        for hit in subDivDict[beatNumber]:
```

```
            mf.addNote(track, channel, hit['midiPitch'], time,
                duration, volume)


# write it to disk
with open("output.mid", 'wb') as outf:
    mf.writeFile(outf)
```