ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Bachelor Thesis

Tangible exploration of sound

Spring Term 2014

Supervised by:
Dr. Stéphane Magnenat
Dr. Francis Colas

Author:
Matthias Bloch

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Tangible exploration of sound

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Matthias	Bloch

Student supervisor(s)

Stéphane	Magnenat
Francis	Colas

Supervising lecturer

Roland	Siegwart

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/associates/students/studium/exams/files-en/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date	Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Symbols

Acronyms and Abbreviations

IR	Infrared
RGB	Additive color space consisting of 3 channels R (red), G (green) and B (blue)
RGB-D	Dataset consisting of RGB and depth data
LWMA	Linear Weighted Moving Average
ADSR	Refers to the phases of an amplitude envelope in sound synthesis

Trademarks and Software

Xbox	Gaming console by Mirosoft
Kinect	Commercial RGB-D camera released for the Mirosoft Xbox
ROS	Robot Operating System
OpenNI	Open source Kinect driver mainly developed by the company PrimeSense
OpenKinect	Community focusing on the use of the Kinect with other devices that the Microsoft Xbox
Libfreenect	Open source Kinect driver built upon contribution from the OpenKinect community
OpenCV	Open source library for computer vision
NumPy	Python library for scientific computing

Chapter 1

Introduction

This thesis on Tangible Exploration of Sound is focused on the development of a system that comprises a RGB-D camera and software to introduce children to the topic of sound in a playful way. The basic idea is to allow the user to sculpt a sound wave in a tangible way, by using physical objects as building blocks. The objects are arranged on a predefined area on a modelling surface, to form the representation of a tone in the time-frequency domain. Two marked axes thereby serve as a reference for time and frequency. The height of the wave model should vary the loudness of the sound.

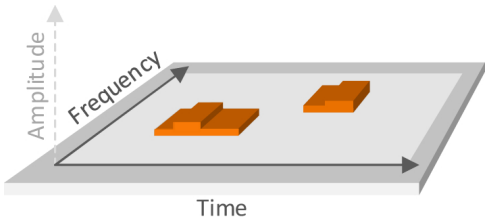


Figure 1.1: Modelling of a sound wave with physical objects.

Above the modelling plane, a RGB-D sensor is installed which captures the geometry of the sound model and provides an elevation map of the scene for the sound computation.

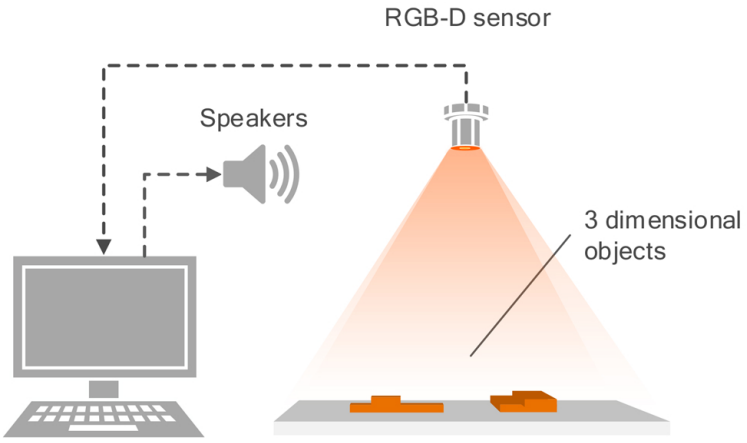


Figure 1.2: Illustration of the modelling scene.

Children can benefit from this experiment in multiple ways, depending on their level of knowledge. Sound and waves are topics quite difficult to teach in a simple way. The proposed system of a tangible sound modelling hereby may serve as a valuable illustration tool.
By modifying their sound model, children experience how sound can change and therefore what defines a tone. At the same time, they learn the methodical procedure in a scientific experiment.
More advanced students should be able to understand the wave character of sound and the concept of frequency and amplitude. Typical phenomena such as superposition or interference can also be illustrated. The modelling in the time-frequency domain in combination with the resulting sound can also help to better understand the concepts of functions and graphs in mathematics.

The data processing and sound synthesis in this project is implemented in the programming language Python. In the following discussion, some direct links to the code of the software are provided. For a more detailed introduction to the software, the repository hosted on GitHub [1] can be visited.

[1] The software repository of the project is hosted at `http://github.com/ethz-asl/sound_sculpt`

Chapter 2

RGB-D camera

For the implementation of this project, a Microsoft Kinect for Xbox was used.

The Kinect sensor is a consumer-ready RGB-D camera which was released in November 2010 by Microsoft for motion tracking and pose estimation in video games. Subsequently, open source drivers have emerged, which allow access to the depth and rgb data stream of the device. There are: The libfreenect driver, built upon contributions from the OpenKinect community and OpenNI, which was mainly developed by PrimeSense, the company designing the Kinect depth sensor. They made way for a broad field of applications beside the Microsoft Xbox gaming console, for which the device was originally designed.

2.1 Specifications

Figure 2.1: Core components of the Microsoft Kinect

The Kinect consists of several core components (Figure 2.1): a RGB camera [1], an infrared emitter [2] and a corresponding receiver [3]. The device is also equipped

with an array of microphones and a built-in tilt motor which allows to adjust the field of view by ± 27 degrees.

Property type	Specification
Color resolution	1280 x 960 pixel
Depth resolution	640 x 480 pixel
Frame rate	30 frames per second
Horizontal field of view	57 degrees
Vertical field of view	43 degrees

Table 2.1: Microsoft Kinect sensor specifications [1].

2.2 Depth resolution

The usual operating range of the Kinect in this project is around 0.6-0.9 m, which is at the lower limit for the sensor's depth estimation. At this height, the resolution of the depth image is approximately 2 mm. Considering the errors of the depth measurement (see section 2.3), the modelling objects should have a height of approximately 2 cm or more to provide good contrast to the base plane (illustrated in Figure 2.3).

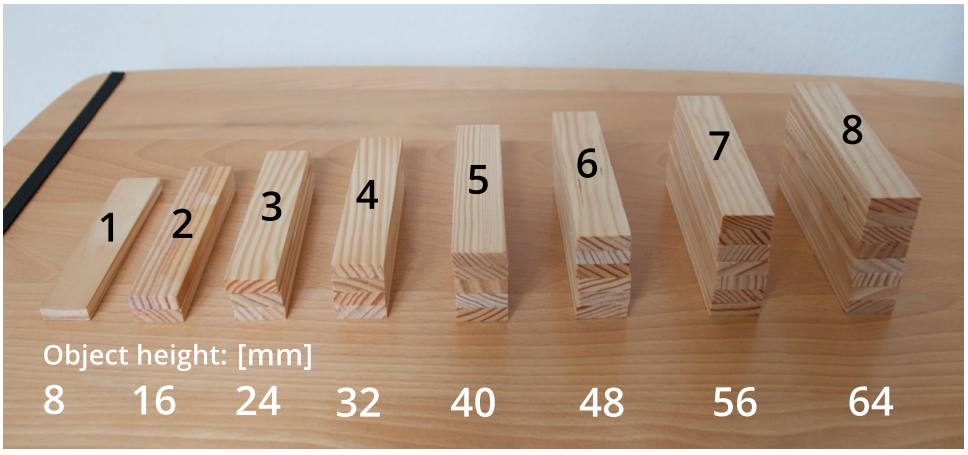


Figure 2.2: Object test setting to visualize the sensor resolution.

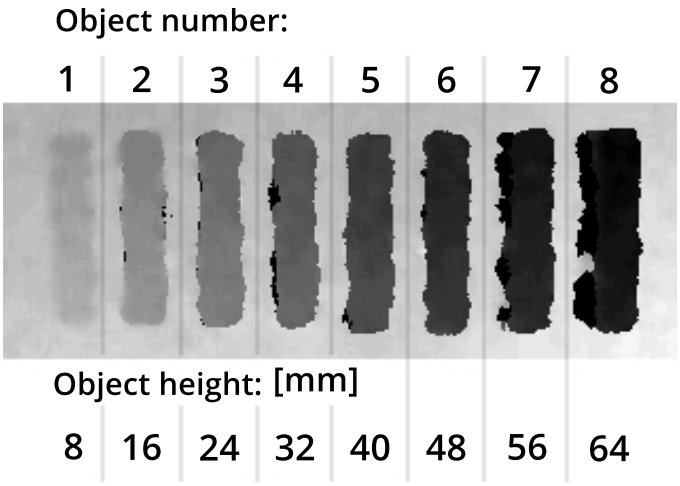


Figure 2.3: Depth image of the test setting in Fig. 2.2. From the third object, with a height of 24 mm, upwards the cuboid objects have clear contours.

2.3 Measurement errors and missing data

If the Kinect fails to estimate a distance, a NaN (i.e. "Not a Number") notification is received at that pixel instead of the depth value. This way the spatial mapping of the depth data is preserved even when some parts are be missing. In the context of this project, the errors can be divided into categories: random errors, material, geometry and device induced errors.

Random errors

The depth measurements in the context of this project take place in the near operating range (0.6-0.9 m) of the Kinect sensor where random errors are not dominants. By averaging the depth measurements, the sporadically occurring random errors can be corrected.

Texture induced errors

In order to capture distance data of a scene, the IR emitter of the Microsoft Kinect creates a diffraction pattern on the modelling scene which is captured by the infrared camera. When mapping reflective (see Figure 2.4) or very absorptive surfaces, the Kinect may have problems registering the depth values correctly as the IR pattern gets disrupted.

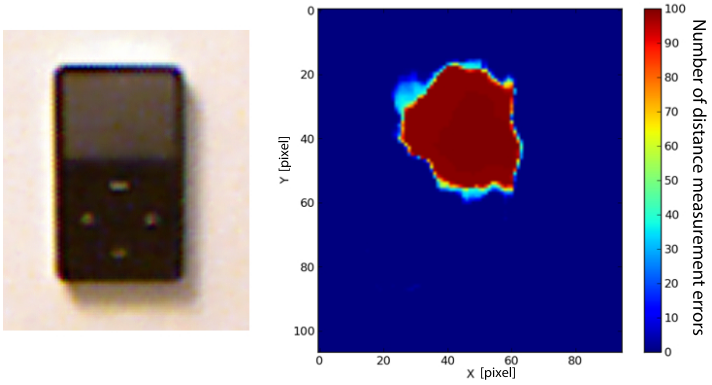


Figure 2.4: Error accumulation in 100 measurements of a reflective surface.

Geometry induced errors

Due to the offset of the IR emitter to the IR camera, some parts of the scene geometry might not be in the range of view of the infrared laser and therefore will not be illuminated although they are visible from the camera perspective. This creates shadow effects in the derived depth image which accumulate around objects.

Figure 2.6 shows the distance measurement errors during 300 frames around a cuboid on a planar surface.

Figure 2.5: RGB image of the test scene

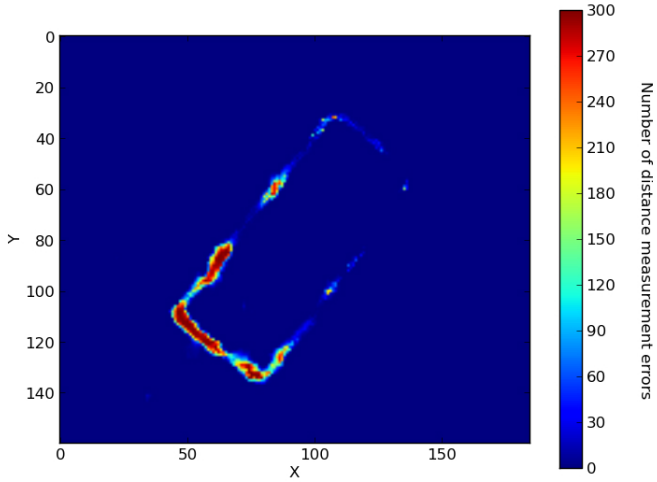


Figure 2.6: Error accumulation in 300 depth measurements

At steep contour gradients errors occur even when using non-glossy materials such as non-laminated paper or unfinished wood. First errors occur approximately at a slope of 2.3 : 1 (i.e. one arbitrary unit in lateral direction and 2.3 units in height).

The critical values are reached at a slope of 3 : 1 and higher. In Figure 2.8 and 2.7 the coherent error locations of 30 measurements are marked in red:

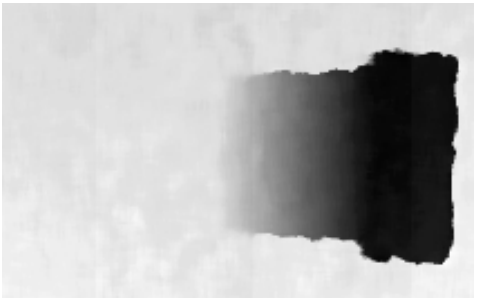


Figure 2.7: Depth image of an unproblematic slope 0.8 : 1

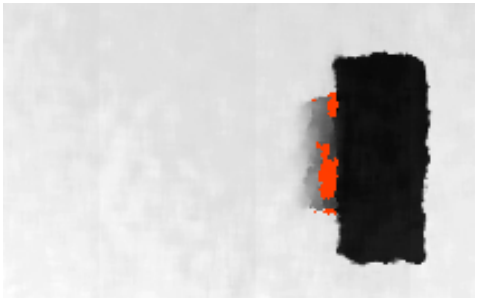


Figure 2.8: Depth map of an object building a slope of 3 : 1 to the camera. Coherent error locations of 30 measurements are marked in red.

Device internal problems

A different type of error is observed when scenes of small distance variations are recorded. Bands of equal width are formed in the depth image:

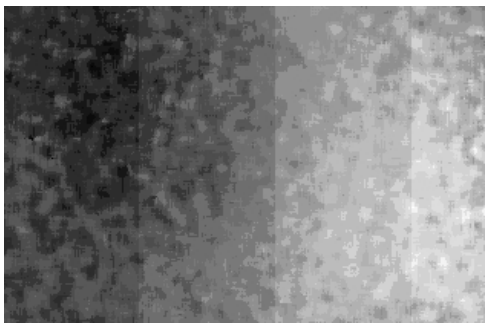


Figure 2.9: Depth map of a planar surface showing artifactual bands

This issue is treated as a device internal problem since these artefacts are stated by Microsoft for the Kinect for Windows [2]. To smooth the band transitions, the images are filtered during the image preprocessing (subsection 4.2.3).

Resulting restrictions

With the camera setting of this project, severe errors mainly occur when analyzing structures with a reflective surface. In order to avoid problem in the depth image acquisition, a non-glossy surface should be used as a modelling plane and the objects used for the wave modelling should also not be too shiny.

2.4 Camera calibration

The most dominant effect to be corrected for is the shift of the depth image relative to the RGB image, resulting from the approximately 2.5 cm separation distance between the IR and the RGB lens. This is necessary for the visualization of the results of the depth analysis in the RGB image, as described in section 4.5. The calibration of the camera is not necessary in all cases. The OpenNI driver for example already comes with a predefined calibration. For the manual derivation of Kinects intrinsic parameters, the RGBDemo software by Nicolas Burrus [3] provides a set of tools. The calibration data can be serialized in a YAML-file which is loaded from the OpenNI or freenect camera driver in ROS.

A manual alignment of the RGB and the depth stream can also be achieved by applying an affine transformation. In this context, the mapping from the RGB frame to the IR frame $RGB \rightarrow IR$ consists of a linear transformation and a translation:

$$\begin{bmatrix} x \\ y \end{bmatrix}_{RGB} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}_{IR} + \begin{bmatrix} c_{31} \\ c_{32} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{31} \\ c_{21} & c_{22} & c_{32} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{IR} \tag{2.1}$$

To solve this system for the matrix coefficients c_{ij}, three pairs of point coordinates $(P(x,y)_{RGB}, P(x,y)_{IR})$ are needed. A manual way of extracting the reference points is to select feature points from the RGB image and the corresponding locations in the depth image (illustrated in section A.4). The transformation matrix consisting of the coefficients c_{ij}, is calculated using OpenCV and stored in a configuration file for further use.

Chapter 3

Camera mount

For the suspension of the camera, an adjustable laboratory mount was used, which enables an almost unrestricted camera field of view.

As the Kinect is intended to be placed on a table in front of the TV screen, it does not have any screw thread to install it on a suspension tool. To avoid modifying the camera, a 3D printed part (for more details see section C.1), adapted to the geometry of the camera base, was manufactured.

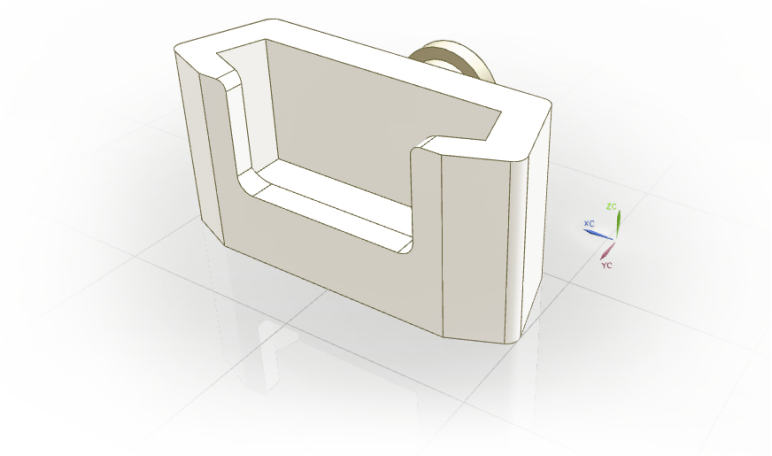


Figure 3.1: 3D model of the camera mount

The support part is connected to the base mount over a releasable clip.

Figure 3.2: Lateral view of the Kinect camera mounted on the holder and the suspension arm

A screwing clamp, which can be easily installed onto a table plate, builds the base of the metallic camera suspension arm and provides enough stability to hold the Kinect camera in its position. Several ball joints allow to adjust the height and orientation of the suspension arm for a flexible positioning of the camera.

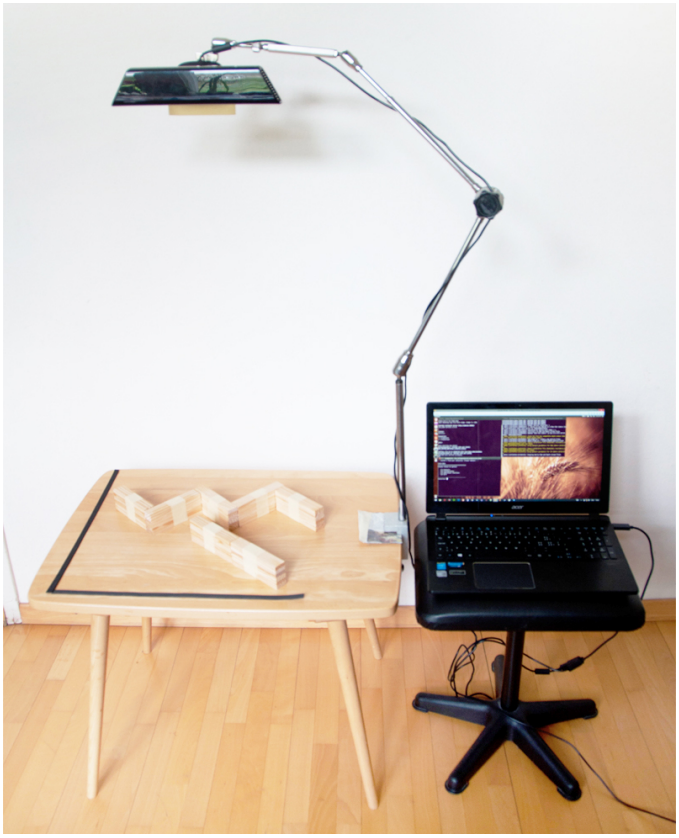


Figure 3.3: Overview of scene, camera and notebook for control and recording

Chapter 4

Software

For the realisation of the software, the following additional Python libraries have been used:

- NumPy - Array processing and computation
- Matplotlib - Plotting
- OpenCV - Visualization and image processing
- PyAudio - Python bindings for PortAudio, an open-source, cross-platform audio library

The robotics framework ROS provides a set of utilities through supplementary packages to interface with the RGB-D sensor:

- Libfreenect based on OpenKinect - Kinect driver
- OpenNI - Kinect driver
- cv_bridge - ROS message conversion

4.1 Image acquisition

The Kinect camera is connected to a computer via USB. For the signal acquisition either the Libfreenect or OpenNI driver can be used, which are both available as ROS packages. The driver converts the initial disparity signal from the Kinect to a depth signal in meters, which is published in ROS as an image message together with the uncompressed RGB data. For efficient processing, the RGB image, as well as the depth data, is transferred into 2D NumPy arrays of shape 640x480, the resolution of the IR camera (table 2.1).

4.2 Image preprocessing

The acquired data from the Kinect sensor undergoes the following preprocessing steps:

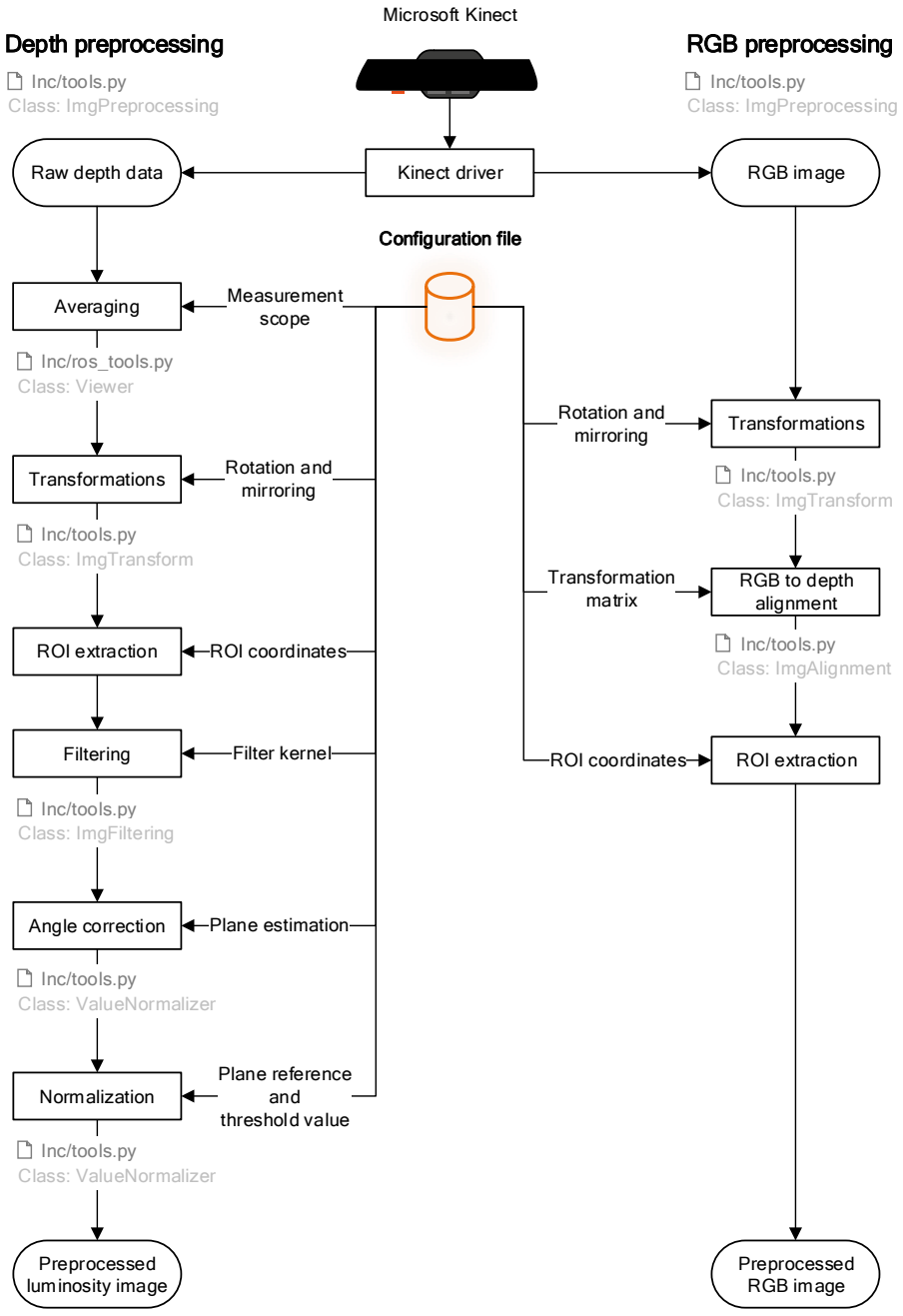


Figure 4.1: Image preprocessing workflow with references to the software modules.

The settings for the preprocessing are centralized in a configuration file and loaded upon initialization of the process.

From the incoming depth data stream from the Kinect driver, a set of depth measurements is recorded and averaged. If the camera is mispositioned with respect to the direction of the reference axis marked on the modelling plane, rotation and mirroring transformations are applied to the acquired elevation map. The region of interest, which represents the modelling region, is derived from coordinates stored in the configuration file. The extracted part of the data is then filtered to compensate for the resolution of the depth measurement. Subsequently, a correction for an offset angle of the camera viewing plane to the modelling plane is applied. The analyzed volume is additionally limited in height by setting a threshold range for the distance measurements. For further calculations, the extracted depth data is normalized.

As the structure analysis (section 4.3) does not rely on the RGB data, the preprocessing of the color image consists only of the main camera frame transformations. If the Kinect driver is not configured, the RGB image additionally needs to be aligned with the depth measurements, for instance through an affine transformation, as discussed in section 2.4.

4.2.1 Depth averaging

In order to increase the signal to noise ratio of the depth data, a set of measurements from the Kinect sensor is averaged. With an approximate frame rate of 30 frames per second, the size of the set is chosen such that it contains around 20 individual distance estimations so the further proceedings are not delayed too much. Besides reducing noise, averaging of the depth signal also compensates for random measurement errors, discussed in section 2.3, and small temporal deviations of the signal.

By normalizing the distance data of a plane recording, a grayscale image is formed with luminosity values relative to the depth measurements. The influence of the signal averaging can hereby be visualized:

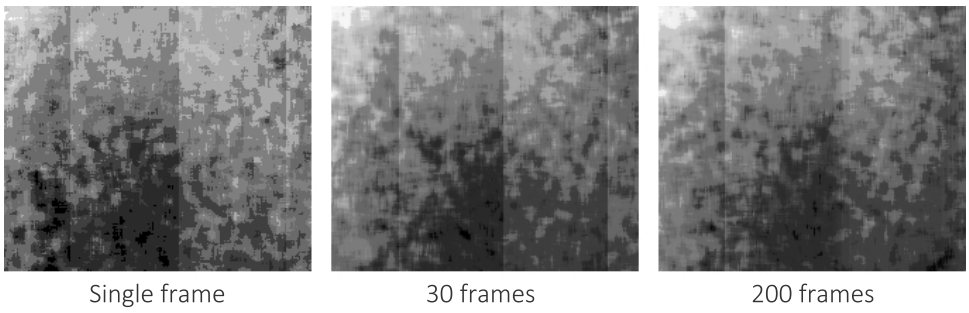


Figure 4.2: Averaged depth images of different set size.

With increasing set size, the resolution pattern of the IR sensor is smeared, building a more reliable elevation map.

4.2.2 Region of interest

The region of interest consists of the modelling area, which is enclosed by the marked reference axis. The edge point coordinates of this rectangular area, which are determined by a manual user selection, are stored in a configuration file. This allows cropping the camera stream to the same shape in every analysis cycle.

4.2.3 Filtering

The raw depth data of the Kinect camera is corrupted by speckle noise with a standard magnitude in the order of 0.002-0.006 m. If a section of the IR data is normalized, the depth measurements can be displayed as a luminosity map, where the errors are visible in the form of a spot pattern on areas of coherent height.

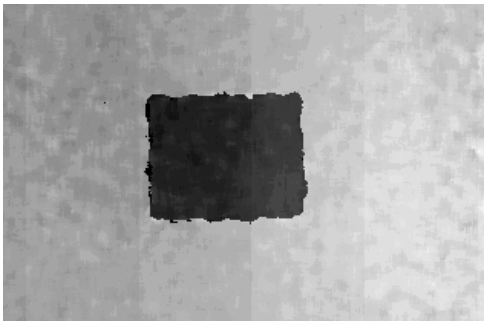


Figure 4.3: Depth image of a cuboid on a flat surface

To smooth out the deviations, bilateral filtering is applied which can be expressed as [4]:

$$h[n] = \frac{1}{w[n]} \sum_{\xi \in \Omega} f[\xi] \cdot c(\xi, n) \cdot s(f[\xi], f[n]) \tag{4.1}$$

With the weight normalization term $w[n]$:

$$w[n] = \sum_{\xi \in \Omega} c(\xi, n) \cdot s(f[\xi], f[n]) \tag{4.2}$$

The filter consists of a range-dependent kernel $c(\xi, n)$ which is combined with an intensity-dependent kernel $s(f[\xi], f[n])$. The filter output $h[n]$ of a pixel in the image f at location n therefore depends on the weights and additionally on the intensity closeness of the neighborhood Ω.

With fitted parameters (table 4.1), derived with an additionally programmed filter testing interface (see section A.3), this filter does not blur at intensity borders and therefore, in contrast to a normal Gaussian filter, preserves object contours in the depth image (Figure 4.5).

Gaussian filtering	
Kernel size	5 pixel
Standard deviation	3.8 pixel
Filter order	3
Bilateral filtering with Gaussian weighting	
Neighborhood size	7 pixel
Depth standard deviation	0.007 m
Spatial standard deviation	2 pixel
Filter order	3

Table 4.1: Robust filter settings.

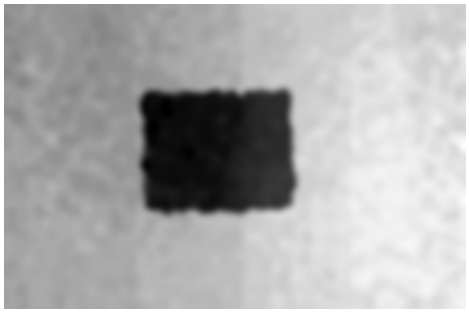


Figure 4.4: Gaussian filtering (Tab. 4.1) applied to Fig. 4.3 using OpenCV

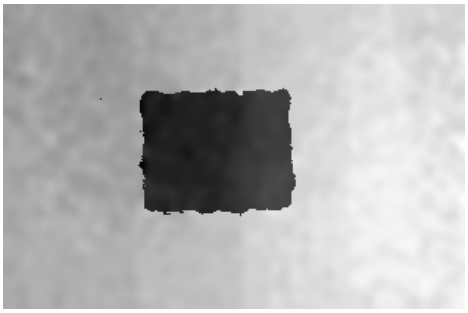


Figure 4.5: Bilateral filtering (Tab. 4.1) applied to Fig. 4.3 using OpenCV provides enhanced object contrast

4.2.4 Plane estimation

A slight tilting angle of the Kinect sensor results in a gradient error in the depth image. To correct for the misalignment between the camera view plane and the modelling plane, an estimation of the recorded surface is made.

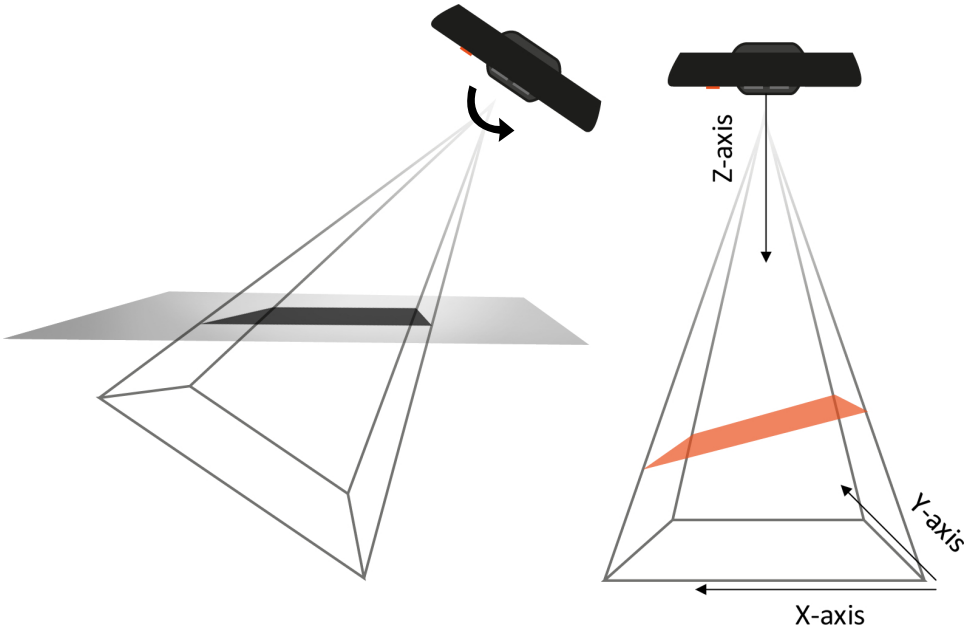


Figure 4.6: Intersection of the modelling plane with the simplified viewing frustum of the camera.

Each of the intersection plane points have to satisfy the general form of the plane equation:

$$a \cdot x + b \cdot y + c \cdot z = d \tag{4.3}$$

Where (a, b, c) denotes the plane normal vector and (x, y, z) the plane point coordinates. Equation (4.3) is parametric in d which is an arbitrarily chosen non-zero constant. A linear equation system is established by using 3 pixels P_1, P_2, P_3 of the untouched depth image, ideally edge points, as reference points for the plane. The equation system expressed in the matrix form $A \cdot x = b$ reads:

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = d$$

The system has a unique non-trivial solution if all equations are linearly independent and matrix A has full rank. This applies if the reference points on the depth image do not form a straight line.

By inverting matrix A, the system can then be solved for the constants a,b,c:

$$\begin{aligned}\begin{bmatrix} a \\ b \\ c \end{bmatrix} = A^{-1} \cdot d &= \frac{1}{\det(A)} \cdot \mathrm{adj}(A) \cdot d \\ &= \frac{\begin{bmatrix} y_2z_3 - z_2y_3 & z_1y_3 - y_1z_3 & y_1z_2 - z_1y_2 \\ z_2x_3 - x_2z_3 & x_1z_3 - z_1x_3 & z_1x_2 - x_1z_2 \\ x_2y_3 - y_2x_3 & y_1x_3 - x_1y_3 & x_1y_2 - y_1x_2 \end{bmatrix}}{x_1y_2z_3 + y_1z_3x_3 + z_1x_2y_3 - x_3y_2z_1 - y_3z_3x_1 - z_3x_2y_1} \cdot d\end{aligned} \tag{4.4}$$

The modelling plane is thus fully described by a point $P(x, y, z)$ and the derived normal vector (a, b, c).

To align the pixels of the depth image along the viewing direction of the camera, each pixel is corrected by an offset depth value δ_z which is calculated using the general plane equation (4.3) and the plane normal (a, b, c):

$$\delta_z[x, y] = z_{ref} - z_{plane}(x, y) = z_{ref} - \frac{d - a \cdot x - b \cdot y}{c}$$

The camera is set as the coordinate origin because the depth measurements relate to the distance to the camera. z_{ref} denotes the reference level for the correction in z-direction, which is chosen as the highest intersection edge point of the calculated plane with the viewing frustum of the camera.

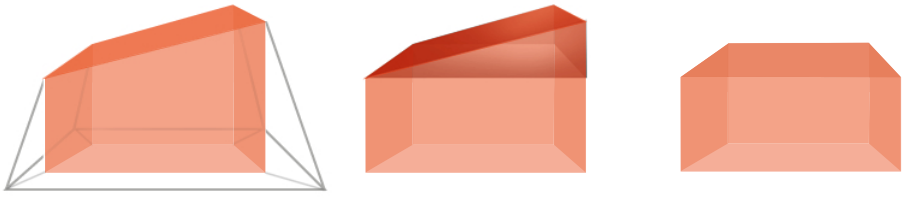


Figure 4.7: Visualization of the plane correction.

The corrected depth map $d[x, y]$ then reads as:

$$d[x, y] = d_{original}[x, y] + \delta_z[x, y]$$

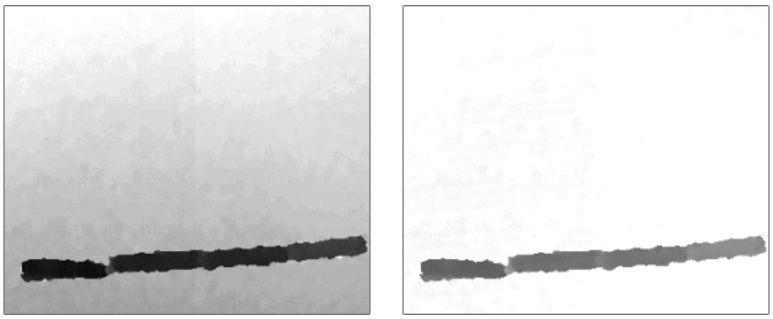


Figure 4.8: High contrast depth image with a gradient error (l) and the corrected image (r)

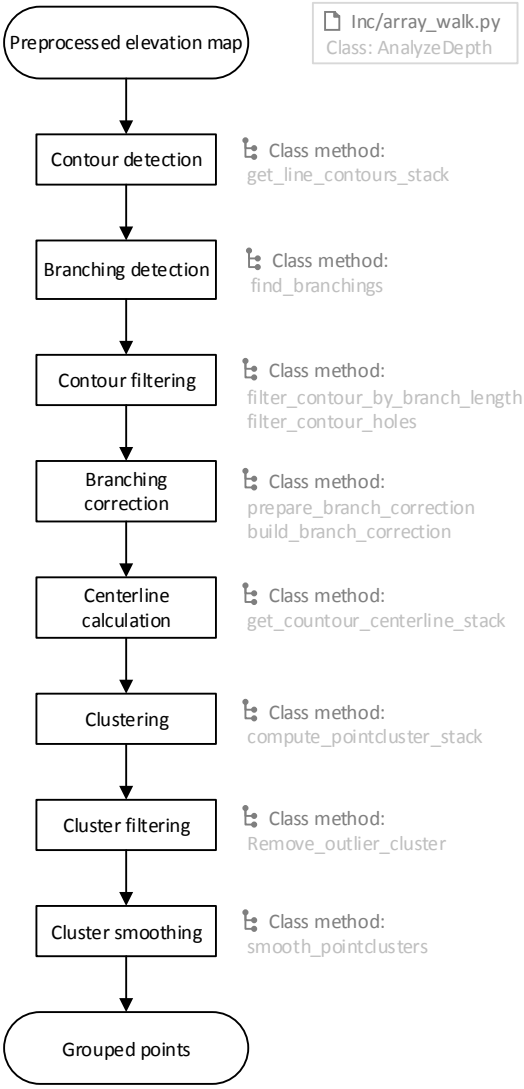
4.2.5 Thresholding and normalization

At this point, the processed distance data still has the unit meter. For further computations, the analyzed volume is additionally limited in height and linearly scaled to a range of 0 to 1. The upper distance threshold to the camera is set to the most distant plane point in the image. This reference value is retrieved from the configuration file. The lower limit is determined by a predefined maximal object height which is also stored in the settings. In the new data format a value of 0 therefore indicates a point in the threshold volume with the shortest distance to the camera, respectively a point of the scene geometry at or above the maximum permitted height. Points lying on the modelling surface or below receive the value 1.

As discussed in section 2.3, the distance data may be incomplete due to measurement errors. During normalization, a value of 1 is assigned to these location to equate them with the modelling surface.

4.3 Image analysis

After the preprocessing, the distance data describing the sound model geometry from the top view, is present in a 2D array with a dimension of 640x480 pixels, matching the resolution of the IR camera. This elevation map represents the surface structure of the volume above the modelling plane.
In this section, a set of geometric object properties is derived from the preprocessed depth image. The shape of the objects are reduced to centerline skeletons which are equidistant to the object boundaries in vertical direction.
In a first step the contours are extracted by setting a threshold for the normalized depth value. Bifurcations and modelling related problems are case specific and have to be handled separately. From the corrected depth transition points, the centerline points are calculated , and are subsequently grouped in such a way that continuous lines of points are formed for subsequent sound synthesis described in section 4.4. To compensate for structural noise, the point sequences are smoothed with a Gaussian filter.



4.3.1 Contour detection

From the elevation map, object contours are derived by introducing a Smitt-triggered depth threshold. The entries of each column of the depth image are compared with the defined threshold range. If the depth value passes the upper threshold for the first time in the current row, the startpoint of an object has been found. When the elevation drops again below the lower threshold, the object ends and the row indices of the transitions are stored. The depth values are again compared to the upper threshold to find the beginning of a new object. This cycle is repeated until the end of the column is reached.

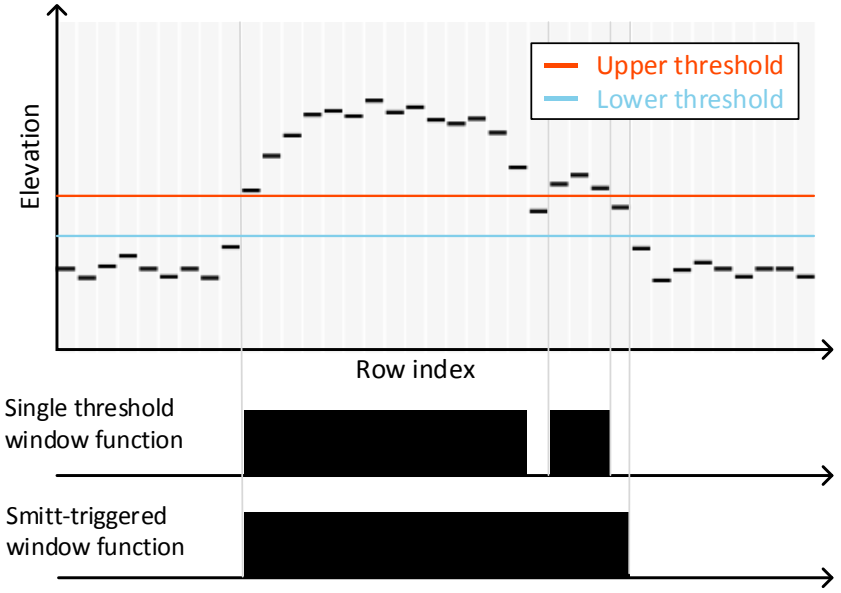


Figure 4.9: Illustration of a Smitt-triggered thresholding compared to a single value threshold. If a single value is set for the threshold, small variations may initiate a new object.

4.3.2 Contour filtering

Occasionally, minor errors in the contour outline may occur, even with Smitt-triggered depth thresholding.

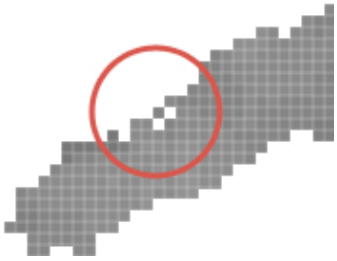


Figure 4.10: Depth image containing outlier pixels resistant to the stepped thresholding.

For this reason the identified contour lines are filtered column-wise by setting a constraint to their mutual separation and the width of the object they describe. Very small contour sections located near a big contour section most likely represent outliers and are therefore removed.

4.3.3 Branch handling

When building more complicated forms with branching structures, bifurcation may cause discontinuity of the centerline calculated as described later in subsection 4.3.5.

To identify the critical regions, the previously derived contour points are examined. In each column of the depth image that contains a branching or merging structure, at least two sets of object start and endpoints must be present that touch the same neighbor contour section.

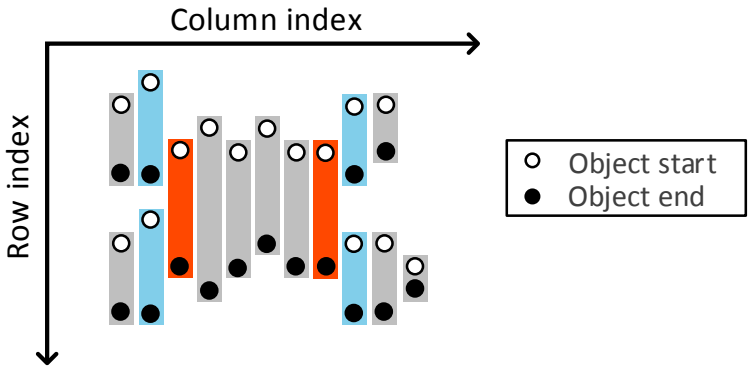


Figure 4.11: Branching and merging structures can be detected by comparing the range of the start and endpoint tuples, derived in subsection 4.3.2. At a bifurcation a parent contour section (marked in red) is present that overlaps at least two child contour sections (blue).

For each column of the depth image the border point tuples are compared to the ones in the neighbor columns.

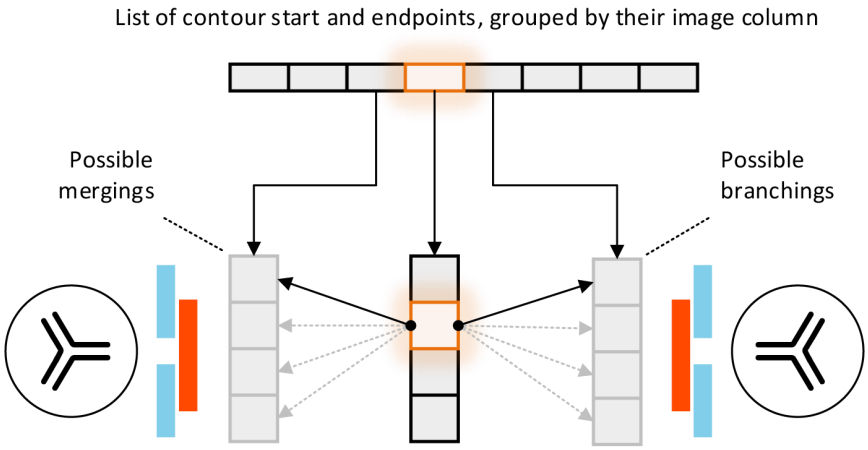


Figure 4.12: For each contour slice in each column, range overlapping contour slices in the neighbor columns are derived which are possibly involved in the formation of a branching.

The parts of the object geometry outline involved in the correction are derived step by step. Starting from the bifurcation origin (Fig. 4.12, red), the adjacent neigbor contour slices are collected in both directions over a predefined range. During the centerline calculation, instead of the normal centerline, direct connections between the ends of the determined contour parts are drawn, which forms a continuous transition between the branches and their origin.

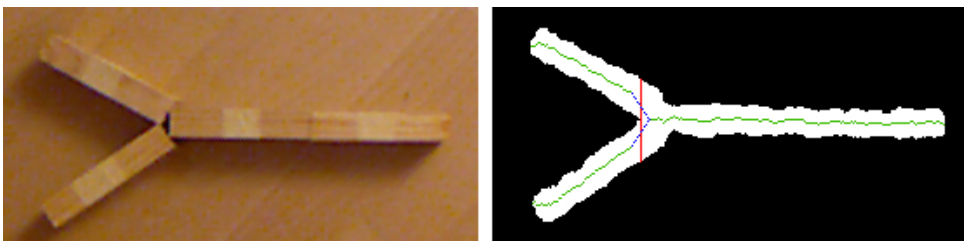


Figure 4.13: RGB image of a branching formation (l) and the derived contour binary map (r). The calculated centerline is marked in green and the branching correction of 10 pixel length in blue. The center of the correction at the identified origin of the branching is indicated by a red line.

4.3.4 Handling modelling errors

Object chaining

If modelling objects are lined up at an angle and the connection is not form-fitting, small side branches are developed due to the unidirectional rasterized contour detection. These branches are found by examining again the branching locations derived

in subsection 4.3.3. Eligible branches of a bifurcation must have a minimum horizontal width to represent a sound of considerable duration.

Figure 4.14: RGB image (l) and the corresponding contour binary map(r). The detected branching locations are marked in red. The unintended branch of the sound model is colored in blue.

Holes

Modelling with non-deformable material may lead to the unintended formation of small holes and gaps between the objects. For example, if multiple objects are placed to build a bifurcation.

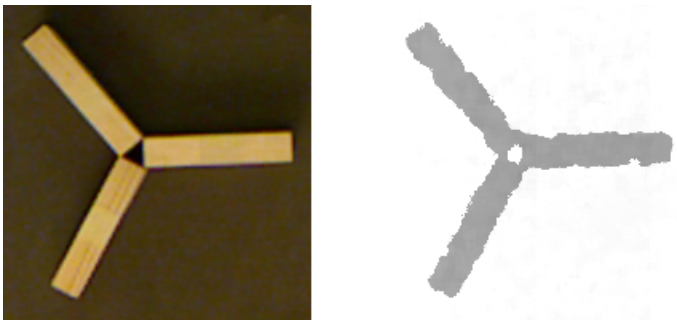


Figure 4.15: Arranging cuboid building blocks to a bifurcation (l) causes the formation of a hole in the depth image (r).

The maximum dimensions of illegal holes is set by defining boundaries for the height and length. To detect holes which satisfy these conditions, in a first step, pairs of branching and merging locations are formed that are separated within the range of the maximum hole width. For each branch of the possible hole origins, the contiguous contour slices in horizontal direction are determined until the position of the allocated merging is reached. Starting from one of the contour lines in the origin of the hole, always the centrally located touching contour slice is selected and vice-versa for the opposite branch on the other side.

If two adjacent branches of the same origin end exactly side by side in the associated merging, they form a closed hole. The contour slices involved in the formation of holes are collected in a buffer and additionally checked for the dimensional restrictions. In order to close these holes, the involved contour lines located at the top are connected with the ones lying beneath, building a single contour slice.

4.3.5 Centerline calculation and clustering

From the processed set of contours, the center points in y direction are calculated. Instead of creating a binary map, the position of the points and the associated luminosity values are stored seperately to allow the allocation of values independent from the original rasterized elevation map. For non-integer indices, resulting from even contour line height or contour corrections, The depth value is linearly interpolated from the enclosing pixels.

As it is detailed in section 4.4, for a fluent sound playback the formation of individual frequency and amplitude variations is necessary. This is done in the image domain by grouping the centerline points to individual curves which describe the mapping $t \to f, x \mapsto y$. For the centerline points in each image column, starting with the lowest index, the shortest connections to centerline points in the right neighbor column are derived. All line segments $\overline{P_i Q_j}$ between the points P_i in the parent column and the neighbor points Q_j are formed and sorted by length in ascending order. Up to a predefined maximum vertical distance between two sequential points of the same object, the shortest connection is extracted and the segments containing one of the involved points are discarded. This way, the centerline points are chained along the horizontal image axis. These clusters can only contain one point per image column and therefore each branch in the centerline point distribution leads to a new cluster which will represend an individual sound.

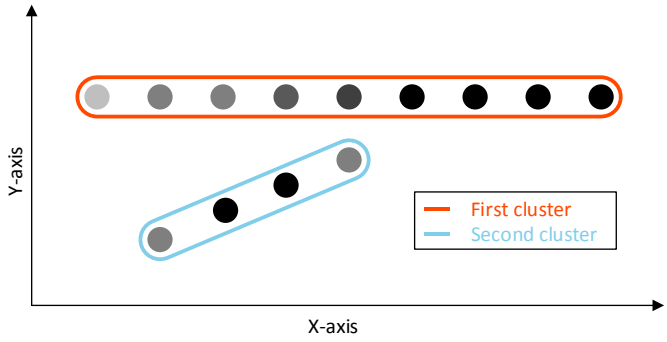


Figure 4.16: Clustered bifurcation structure. Each branch of the point distribution forms a seperate group.

Furthermore, the depth values associated with the point coordinates of the clusters are verified. If the corrections from subsection 4.3.4 are applied, there might be points generated that are located outside of the wave model (e.g., if the centerline crosses a hole). To bridge these inconsistent gaps, a linear interpolation between the depth values of the nearest valid points of the sequence is assigned.

4.3.6 Cluster smoothing

The IR measurement is corrupted by structural noise causing deviations in the calculated centerlines. In order to compensate for the positional noise in vertical direction, the centerlines are smoothed by linear filtering.
To apply a filter, the filter kernel k is convolved with the finite point distribution input f[n]:

$$(f * k)[n] = \sum_{m=-\infty}^{\infty} f[n-m]k[m]$$

Performing the convolution on the initially zero padded input sequence $f[n]$ leads to an edge damping when the kernel is overlapping the sequence. To avoid this effect, the input is extended on both ends by a buffer zone with a gradient calculated off the border region.

Gaussian filter

Assuming that the position is corrupted by a Gaussian noise with zero mean and standard deviation σ, a Gaussian smoothing can be applied. The centered weighting function of the one-dimensional Gaussian kernel can be expressed as:

$$k[n] = \exp(-\frac{n^2}{2\sigma^2})$$

The total weight of the kernel is normalized to 1.

Parameter estimation

The second order central moment can be calculated from the empiric contour distribution as a variance estimator $\hat{\sigma}^2$. For the normal distribution model $Y \sim \mathcal{N}\left(\mu, \sigma^2\right)$, where Y is the random variable of the depth measurement, applies:

$$E(Y) = \mu, E(Y^2) = \sigma^2 + \mu^2 \longrightarrow \sigma^2 = E(Y^2) - E(Y)^2$$

The raw empiric moments, which are defined as $\hat{m}_k = \frac{1}{n}\sum_{i=1}^{n} y_i^k$, can be used to derive the estimator $\hat{\sigma}^2$.

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

Taking into account that the variance estimator $\hat{\sigma}^2$ is biased, respectively asymptotically unbiased, Bessel's correction can be applied by multiplying the estimator with the factor $\frac{n}{n-1}$. The unbiased sample variance [5] is then given as:

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \overline{y})^2$$

For robust filtering, the variance should be chosen $\sigma^2 = 1.7$ or greater. In this case, slight over-smoothing is unproblematic as the characteristics of the filtered lines still remain.

An illustration of the structural variation of depth measurement can be seen in the following top view depth image of a cuboid.

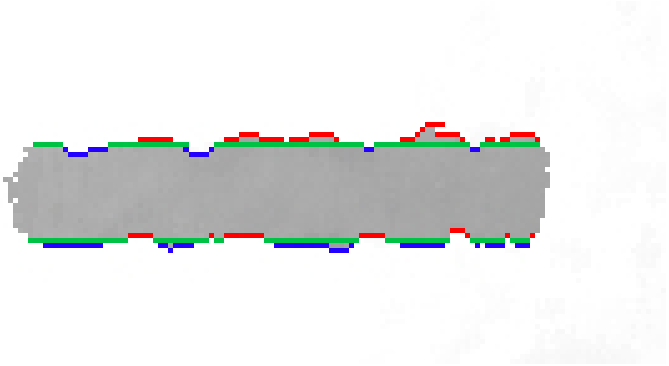


Figure 4.17: Structural noise in the depth measurement of a cuboid contour. Green: Average value, Red/Blue: Deviations

Linear weighted moving average filter

Alternatively a linear weighted moving average (LWMA) filter can be applied where the weighting of the neighborhood, in contrast to a standard box blur, is linearly dependent on the distance from the examined point.

For a discrete input $f[n]$ to a LWMA filter of size N, the filtered values $m[n]$ are given by:

$$m[n] = \frac{2}{n(n+1)} \cdot \sum_{i=1}^{N} i \cdot f[n - N + i]$$

The weighting kernel of odd sample length N can be written as:

$$m[n] = -\operatorname{sgn} n \cdot \frac{2}{(N-1)} \cdot n + \frac{4}{(N-1)^2}$$

Filter comparison

The convolution kernels of a Gaussian and a linear weighted moving average filter with 31 samples are displayed in Figure 4.18. Due to the exponential decay, a Gaussian weighting function theoretically should have an infinite span. However it is appropriate, regarding the minimal impact, to truncate the kernel window for this application.

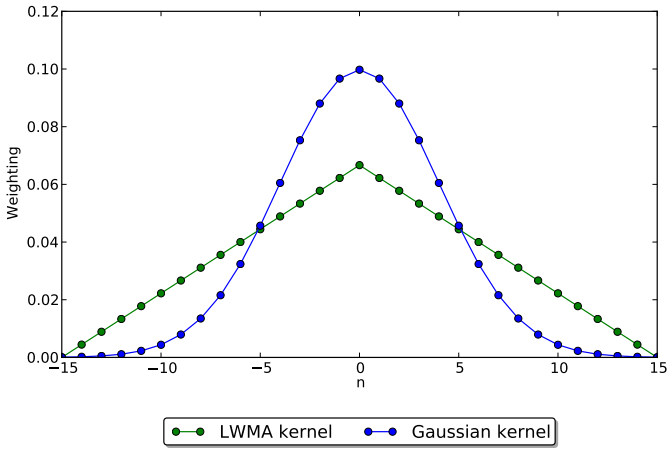


Figure 4.18: 31 sample width 1D convolution kernels. The Gaussian kernel has a standard deviation of 4.

The results of the filter convolution with a point sequence from a measurement is illustrated in Figure 4.19.

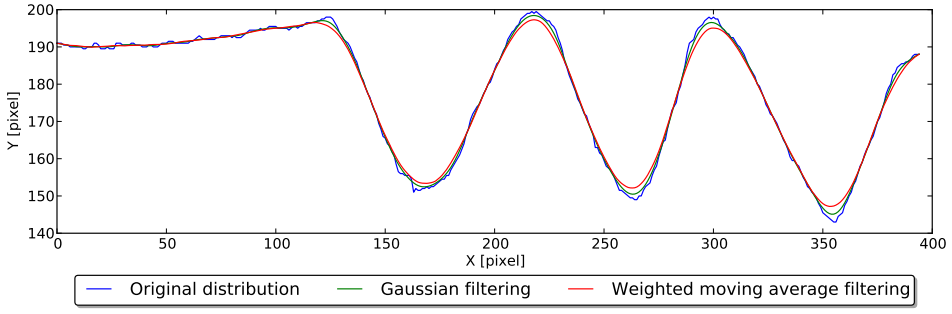


Figure 4.19: Curve regression using the smoothing kernels displayed in Figure 4.18

4.4 Sound synthesis

Before the computation of the audio waves, the spatial properties of the derived point distribution are linearly mapped to sound properties. The vertical image axis is mapped to a predefined frequency range f_{min}–f_{max}:

$$f(y) = \frac{f_{max} - f_{min}}{y_{max}} \cdot y + f_{min}$$

The position along the vertical axis is denoted by y and y_{max} is the image height.

The normalized depth values ρ in the range of 0 to 1, are used as a reference for the sound amplitude A. Down to the lower threshold value α they are likewise linearly mapped to an amplitude range of A_{min} to 1.

$$A(\rho) = \frac{A_{min} - 1}{\alpha} \cdot \rho + 1$$

Since the distance measurements refer to the separation from the camera, the depth values are reversed to the height of the wave model. Accordingly, the highest points have a depth value of 0 and get the maximum amplitude of 1 assigned.

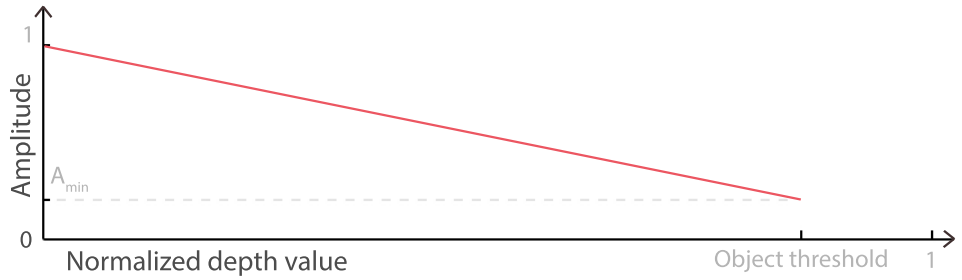


Figure 4.20: The depth values up to a threshold value get linearly mapped to an amplitude range of A_{min} to 1.

The default duration for the sound of a single point is set to a value around 0.03 seconds, which results in a total sound modelling time span of 12 seconds at an average region of interest of 400 pixel width.

In subsection 4.3.5 sequences of centerline points have been formed to allow creating individual waves for each geometrical feature of the wave model. This is necessary to prevent a wave cut-off (discussed in the following subsection 4.4.1) and to apply a transition in the time-frequency domain between points. Each of the point clusters therefore describes a modulation of frequency and amplitude over time from which an individual sound can be generated.

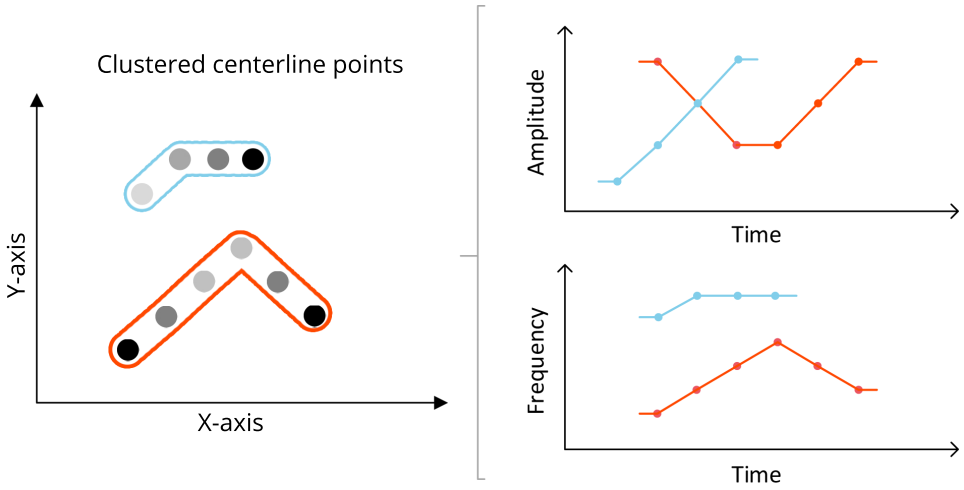


Figure 4.21: Example of mapping two point sequences (l) to wave properties. The stepped amplitude and frequency values are linearly interpolated to build a smooth transition (r).

In the software, the amplitude modulation and the calculation of the oscillation is handled in separate processes. To build the final wave, the amplitude envelopes are multiplied with the oscillator signals.

4.4.1 Oscillators

To produce soft sounding tones, a sinusoidal waveform is chosen for the audio signal. If the object centerline points are not grouped and for each point a seperate oscillator is created, a cut-off is induced if multiples of one half of the wave length do not match the defined per-point time span t_d. This manifests itself as a clicking noise in the final audio output. To prevent this effect on individual oscillations, the duration or the frequency can be adjusted to form closed waves.

The duration of the oscillation t_d can either be truncated or extended to end the wave in 0.

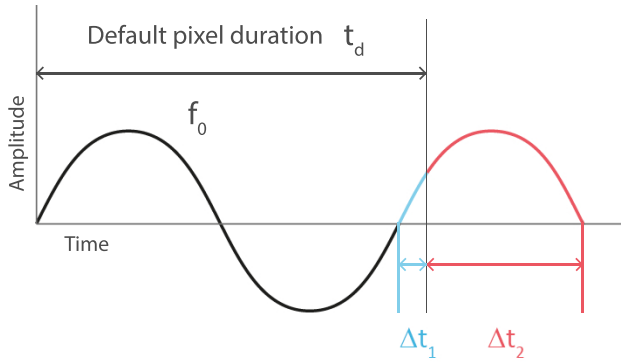


Figure 4.22: Truncation or extension of the wave to 0 leads to a time change $\Delta t_{1,2}$

$$\Delta t_1 = t_d - \frac{\lfloor t_d \cdot 2f_0 \rfloor}{2f_0} \qquad \text{and} \qquad \Delta t_2 = \frac{\lceil t_d \cdot 2f_0 \rceil}{2f_0} - t_d \tag{4.5}$$

For a sine wave this leads in the worst case to a change of duration of almost $\pm\frac{0.5}{frequency}$, half a period of the signal. At a minimum frequency of 20 Hz, which corresponds to the lower threshold of human hearing [6], the temporal change amounts 0.025 seconds.

Instead of changing the duration of the wave, we can also adapt the frequency.

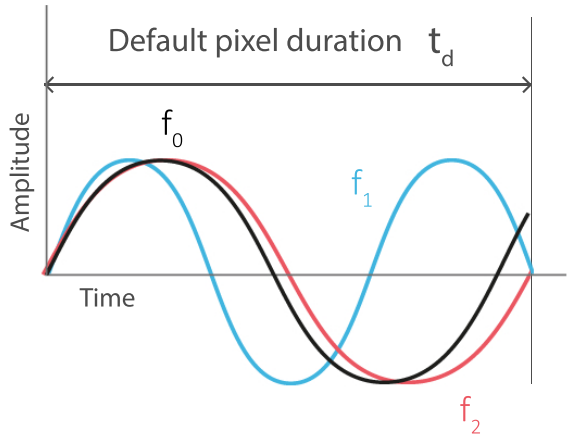


Figure 4.23: Frequency variation to adjust the original wave period $1/f_0$ to end the wave at t_d in 0

$$f_1 = \frac{\lceil t_d \cdot 2f_0 \rceil}{t_d} \qquad \text{or} \qquad f_2 = \frac{\lfloor t_d \cdot 2f_0 \rfloor}{t_d} \tag{4.6}$$

Adjusting single waves entails, as outlined, a noticeable frequency or time error and is therefore unsuitable for precise sound synthesis. It also requires the default duration of the single frequency waves to be set large enough to ensure at least half an oscillation, in case of a sine wave.

By forming groups composed of several temporally aligned waves, oscillations with a stepwise modulated frequency can be generated. For this purpose, the calculated centerline points have already been grouped in the image domain (subsection 4.3.5). From the stepwise linear wave property functions a set of linear oscillators is calculated. The oscillators are sampled $t \mapsto \frac{n}{f_s}$ with the frequency $f_s = 44100$ Hz to create a discrete audio sequence.

By applying phase shifts, the individual oscillators can be joined to a single continuous wave.

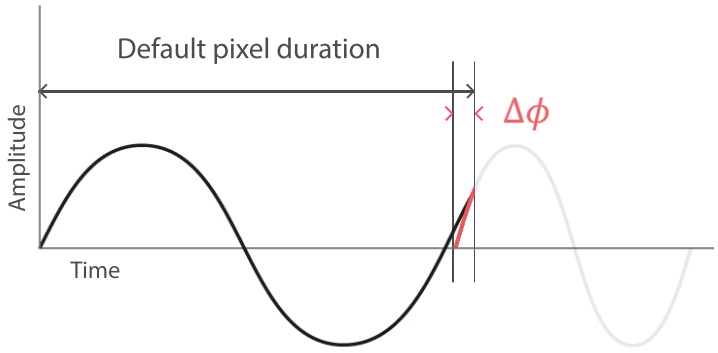


Figure 4.24: Alignment of two waves by applying a phase shift $\Delta\phi$ (4.7)

In order to derive the required phase offsets $\Delta\phi_i$, the oscillators s_i are calculated with an additional length of one sample which is removed again for the accumulation of the signal. The phase shift is then determined from the extension of the previous wave:

$$\Delta\phi_i = \arcsin(s[-1]_{i-1})$$

To continue the following waves in the same direction, an additional shift has to be applied, if the gradient $s_{i-1}[-3] - s_{i-1}[-2]$ is negative. The corrected phase shift can be written as:

$$\Delta\phi = \operatorname{sgn}(s[-1]_{i-1}) \cdot \pi - \arcsin(s[-1]_{i-1}) \tag{4.7}$$

The chained oscillators are calculated as chirp signals [7] (oscillator with a linear changing frequency). For two frequency levels f_1 and f_2 the time dependent transition frequency reads:

$$f(t) = f_1 + k \cdot t \qquad \text{with} \qquad k = \frac{(f_2 - f_1) \cdot f_s}{N} \tag{4.8}$$

The frequency function to a signal $\sin(\theta(t))$ with a time dependent phase function $\theta(t)$ is defined as:

$$f(t) = \frac{1}{2\pi} \cdot \frac{d\theta(t)}{dt}$$

The phase function $\theta(t)$ at the time t' therefore is:

$$\theta(t') = \theta_0 + 2\pi \cdot \int_0^{t'} f(t)\,\mathrm{d}t$$

With the frequency function (4.9) and the initial phase θ_0 chosen corresponding to equation (4.7), the resulting phase accumulation can be written as:

$$\theta(t) = 2\pi \cdot t \cdot (f_1 + \frac{k}{2} \cdot t) + \Delta\phi \tag{4.9}$$

The oscillator sequence $s[n]$ to the sampling vector $n = [0, 1, 2, ..., N-1]$ of length $N = \lfloor t_d \cdot f_s \rfloor$ can now be written as:

$$s[n] = \sin(2\pi \cdot \frac{n}{f_s} \cdot (f_1 + \frac{(f_2 - f_1)f_s}{2N} \cdot \frac{n}{f_s})) \tag{4.10}$$

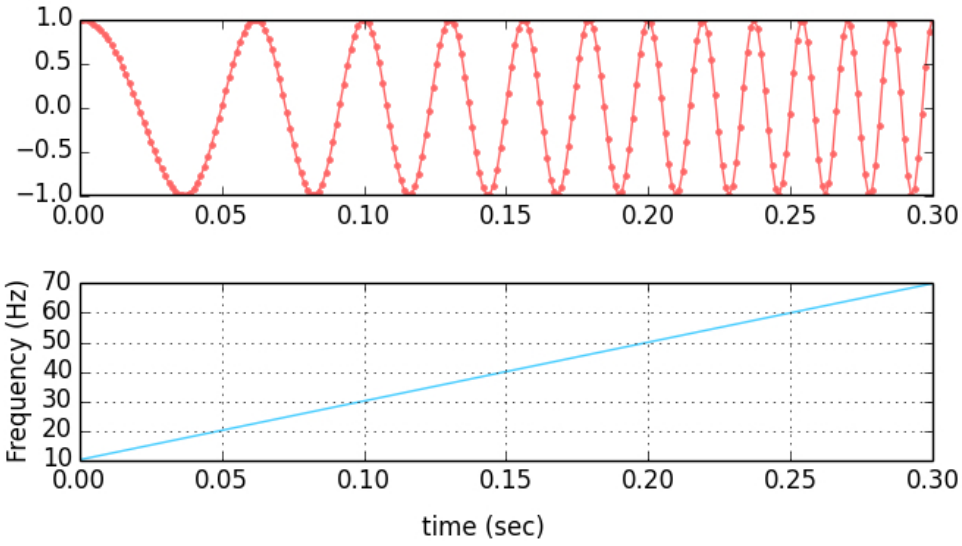


Figure 4.25: Example of a chirp signal with a frequency transition from 10 to 70 Hz in 0.3 seconds, sampled at 800 Hz

4.4.2 Amplitude envelopes

The stepped amplitude values of each point sequence derived in section 4.3 are linearly interpolated to build a smooth envelope for the corresponding oscillator wave. Filtering of the depth values is no longer necessary, as a correction was already applied in subsection 4.2.3.

ADSR envelope

Each amplitude variation is multiplied with a depth value independent ADSR envelope which regulates the overall amplitude in four phases of different length: Attack, Decay, Sustain and Release.

In the Attack phases, the amplitude is linearly increased until the maximum is reached. It follows a linear decay to a constant amplitude level. During the Release phase the amplitude fades out again to 0.

The duration of the Attack, Decay and Release phase are defined as a percentage of the total envelope length which has to match the length of the corresponding oscillator sequence. The corresponding lengths in samples are truncated.

In experiments with different sound models standard settings of 20% Attack, 15% Decay and 20% Release have proven to build a smooth transition between individual tones. During the Sustain phase the amplitude is kept at 0.8.

The resulting piecewise linear ADSR function of 40 samples is displayed in Figure 4.26:

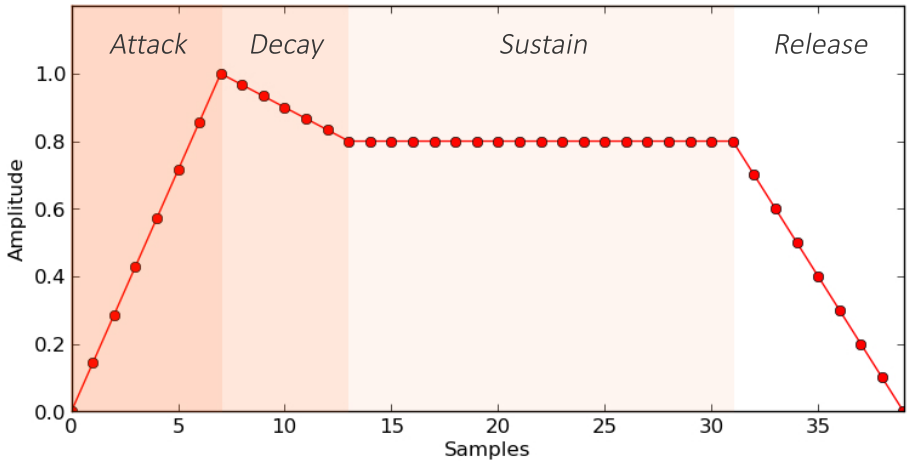


Figure 4.26: ADSR envelope of 40 samples length with 20% Attack, 15% Decay, 20% Release and an amplitude of 0.8 during the Sustain phase.

4.4.3 Output normalization

To combine multiple waves in one audio channel, the superimposed wave data has to be normalized to an amplitude of 1:

$$wave_{comb.} = \sum_{i=1}^{N} \frac{wave_i}{N}$$

If the maximum number of simultaneous sounds N of a wave model is not restricted, the scaling can be applied relative to the current scene.

In order to make the amplitude difference between various object settings comparable, the maximum number of parallel objects N can be predefined.

4.5 Sound playback

For presentation purposes the start of a depth analysis cycle is coupled to the input signal of a USB push-button. After the calculation process is finished, the normalized audio data is transmitted to the sound card.
To allow the user to track the played sound on the input image of the wave model, a live visualization of the time axis is created using OpenCV. The audio playback and the audio visualization are running in two separate threads. The calculated audio data from a depth image is written in chunks of the specified per-pixel-duration to the output stream. After each data slice, the horizontal position is updated and the visualization thread, which acts as a listener, marks the position of the time reference axis in the corresponding RGB image and displays it in an OpenCV window (section A.2).

Figure 4.27: USB push button to activate the analysis process

4.5.1 Looped analysis

A continuous analysis of the scene is implemented through a threaded buffer system. When the loop is started, the audio calculation thread records a sequence from the Kinect depth signal and calculates the audio output data. This data is filled in a buffer stack from which a playback thread fetches the data. As soon as the buffer is empty again, the calculation cycle continues.

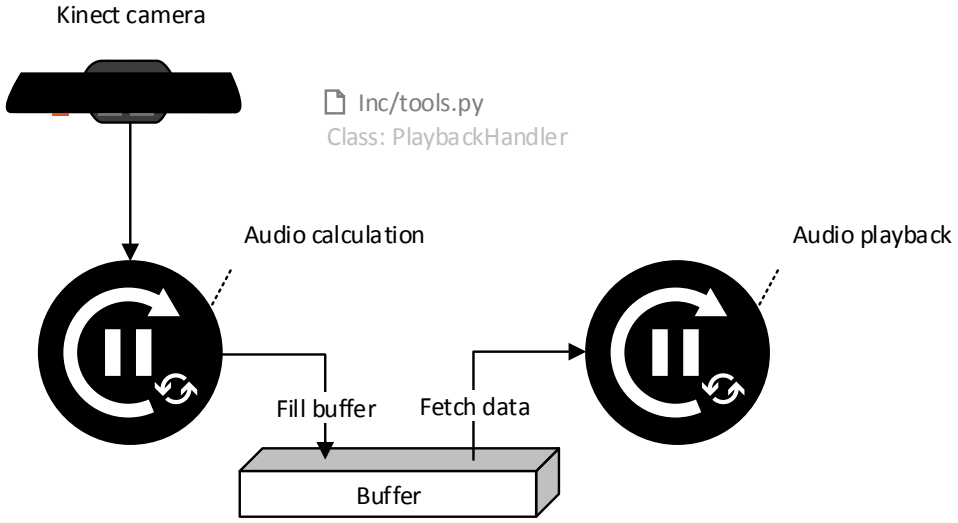


Figure 4.28: Workflow during a continuous analysis and playback cycle

Chapter 5

Usability study

In the course of the thesis a student experiment was conducted to test different aspects of the proposed system: The intelligibility of the sound modelling concept and, on the other hand, the feasibility of a student experiment and the stability of the system itself. The experiment was carried out at the Steigschule Schaffhausen with a small group of four 4th grade students with an average age of about 10 years.

A set of exercises was created (see Appendix B) to introduce the students to the sound modelling concept and to evaluate their understanding. The worksheets are designed so that the children first get to know the sound modelling concept. They have to rebuild certain sounds that only differ in a single property (pitch or volume). The meaning of the horizontal time axis is also discussed. After the introduction the students are asked to formulate the observed properties of sound in their own words. A final exercise is to draw a shape and predict the sound output. This way, the understanding of the modelling in the time-frequency domain is checked again. All children were able to solve these tasks without further help themselves. They could easily classify the properties of different sound models (pitch, loudness and duration) so that they could soon build a simple song they have learned in class. They participated enthusiastically and had a lot of fun during the experiment.

Besides the evaluation of the modelling concept, the goal of the experiment was also to test the stability of the system itself. The transport of the device was not difficult and it could easily be installed onto a table. The configuration did not take long but a connectivity issue of the Kinect driver cost some additional time. After the configuration the device could be operated by the children themselves. With four children participating in the experiment, it was possible to also allow sound modelling as a group. It is not recommended to have more simultaneous participants than this at the same time so that the students still can focus on the topic and they do not have to wait too long until it is their turn to model sound.

Chapter 6

Conclusion

In this thesis, a software library for the use with a commercially available RGB-D camera was developed to analyze three-dimensional wave models consisting of physical objects and to transfer the geometrical properties to sound.

The camera was installed on a suspension arm which allows a good camera field of view, is stable, adjustable and at the same time is also portable. The device can be installed quickly and easily onto any table. The programmed software allows the acquisition, customized preprocessing and filtering of the RGB and depth data from the Kinect camera. A shape analysis algorithm detects objects on a flat surface and reduces their geometry to skeleton lines. During the analysis process, various correctional functions are applied to handle noise in the depth data and problems in sound model geometry for an increased detection rate. Additionally, the skeleton points are clustered to distinguish among different geometric features. For the sound synthesis, the geometrical data is transferred to sets of wave properties. A synthesizer allows to generate sound from scratch and supplies different modules for wave shaping. In order to visualize the audio playback, an animation can be displayed on the computer screen, which indicates the currently played part of the sound model. The configured device can be operated by children themselves. Different modes are available which start the analysis on the signal of a push button or automatically. A user interface provides an additional set of tools and allows a quick configuration and the management of analysis parameters and settings.

The proposed sound sculpting concept is based on an ideal top view of the RGB-D sensor but to obtain a sufficiently high resolution, the Kinect camera needs to be installed at a short distance to the modelling plane (around 0.7 m) which is also the lowest operating point of the depth sensor. At this height, only a certain part of the camera image can be treated as a direct top view and the modelling area is therefore limited. Due to the limited depth resolution of the Kinect, modelling with small objects is not possible.

The experiment with the students however showed that the proposed system for a tangible sound modelling works and that children understand it. In the experiment they have learned about the properties of sound, namely frequency and amplitude, and they also were introduced to the concept of functions and the procedure in a scientific experiment. Therefore, the proposed system may serve as a valuable teaching tool.

A continuation of the project could be done in several directions. Additional geometry or texture parameters could be introduced to enrich the variety of sound. As an example, object colors could be used to modulate vibrator or chorus. From a teaching aspect, adding different wave forms, for example as a function of the object width or the height profile, would be reasonable. A reversed operating mode, where the user can record a sound and visualize it in the time-frequency domain would facilitate the understanding of the wave model. Another possible extension would allow the user to pause the audio playback and to modify the currently played part of the wave model to hear the change in amplitude or frequency directly. In order to enhance the experience of the experiment itself, a projector could be used to visualize the modelling area, the reference axes and the playback directly on the modelling surface.

Acknowledgement

I would like to thank my supervisors Dr. Magnenat and Dr. Colas for their guidance and the pupils at the Steigschule Schaffhausen for their participation in the evaluation of the proposed system for a tangible sound exploration.

Appendix A

User interface

In order to ease the configuration for the analysis, the management of settings and the access to the main functions of project library a text-based user interface was created. The navigation is purely based on keyboard inputs.

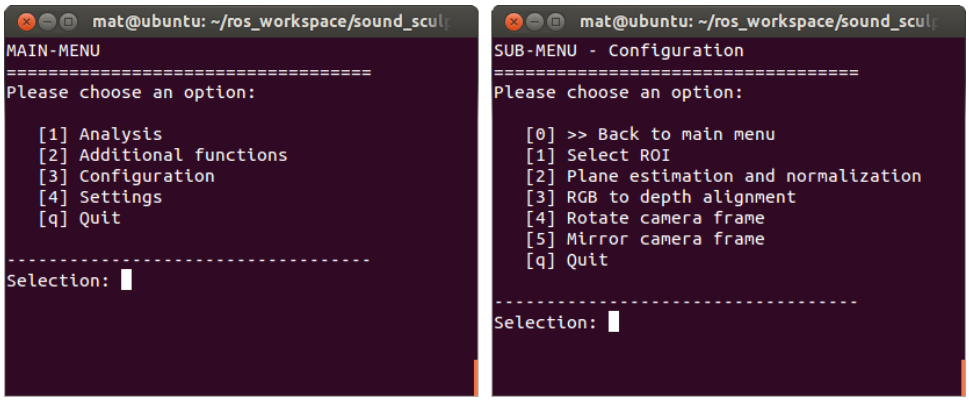


Figure A.1: Text-based user interface

The structure of the interface is illustrated in the following graphic:

Figure A.2: User interface menu structure

A.1 ROI selection

Inc/tools.py Class: ROISelect

A top view capture of the scene is displayed in an OpenCV window. By attaching a callback function to the standard mouse click action, a rectangular area can be easily selected in either the RGB or the depth image

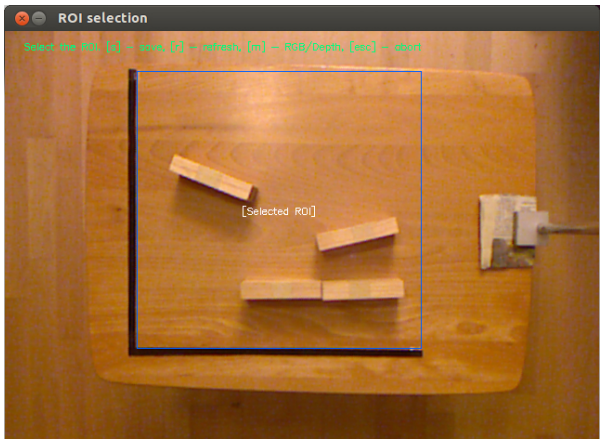


Figure A.3: Region of interest selection

A.2 Sound playback visualization

Inc/tools.py Class: VisualPlayback

The playback animation is initiated by the playback handler and displays the current position on the time axis.

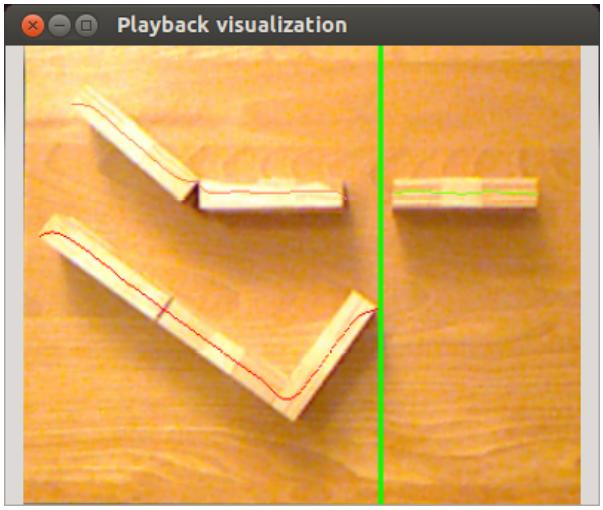


Figure A.4: Audio playback visualization

A.3 Prefiltering

Inc/ros_tools.py Class: ValueReader

To easily read out specific value of the depth image and to test filter parameters an additional interface is created.

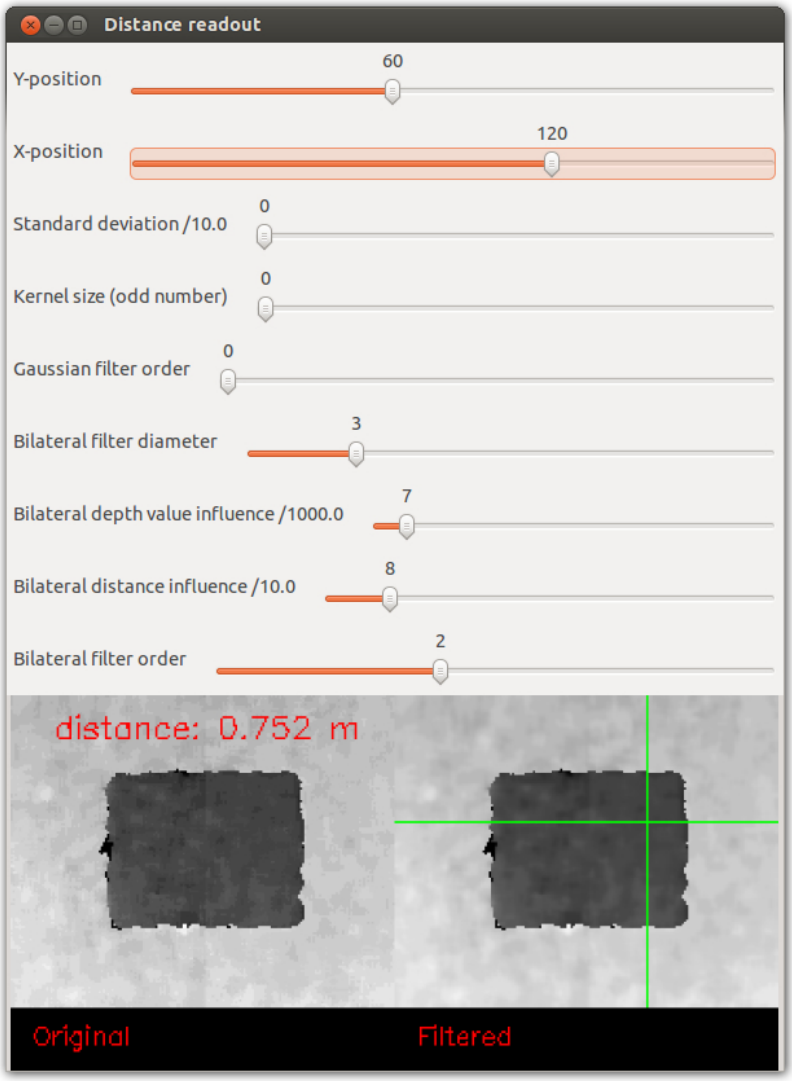


Figure A.5: Interface to test filter parameters and to read out distance estimations from the Kinect.

A.4 RGB to depth alignment

Inc/tools.py Class: ImgAlignment

The reference points for the manual alignment of the RGB image to the depth map can be directly selected on the corresponding images using OpenCV.

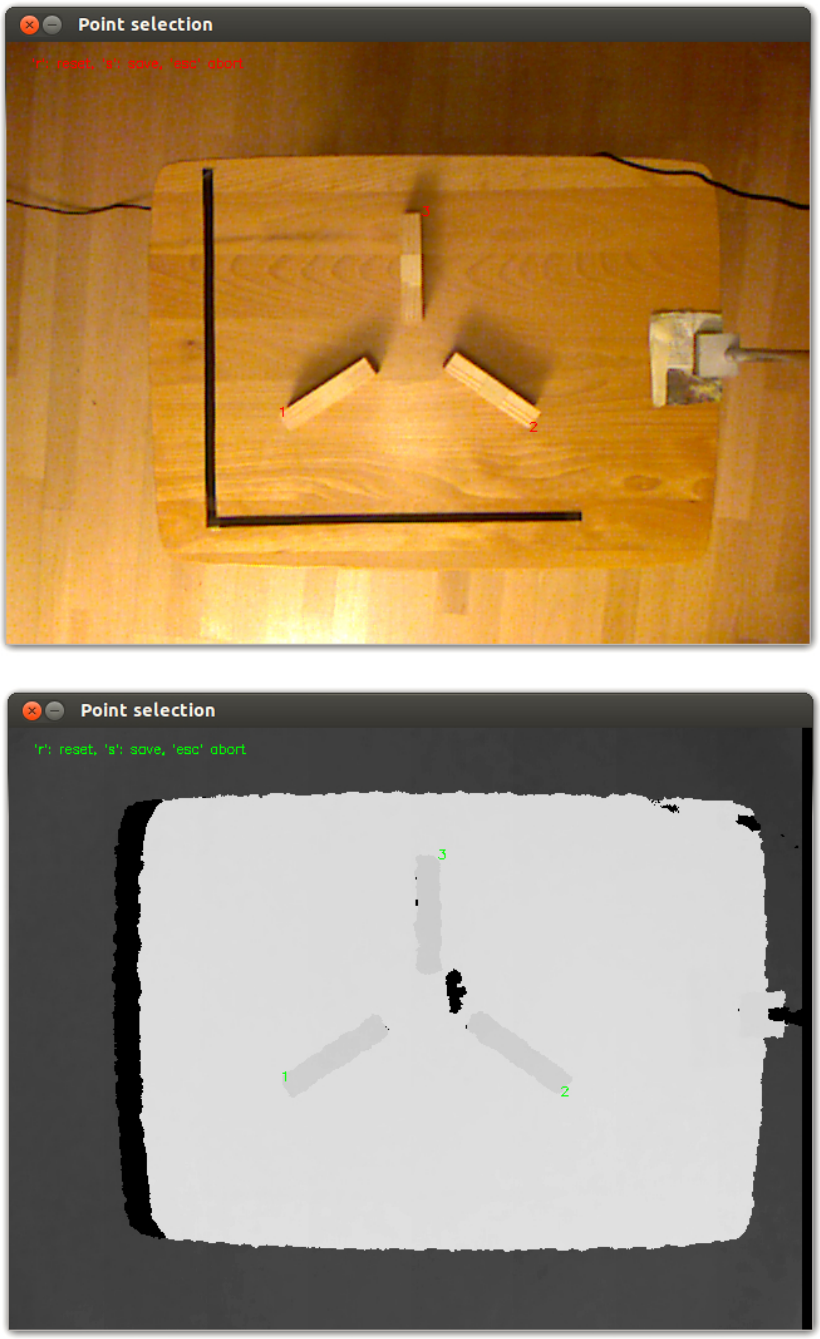


Figure A.6: Manual reference point selection for an affine transformation.

Appendix B

Usability study worksheets

Aufgabenblatt: Modellierung von Tönen

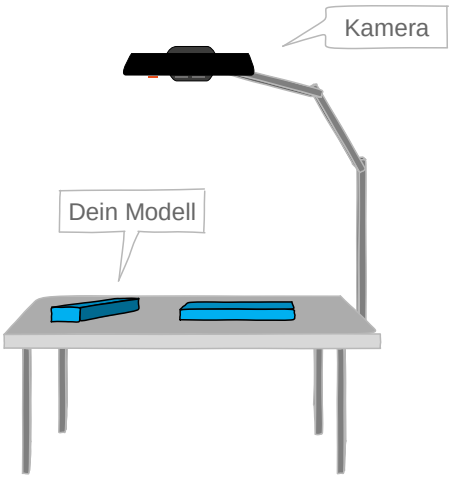
Modellierung von Tönen

Worum geht es?

In diesem Experiment geht es um die Eigenschaften von Tönen, wie wir sie jederzeit hören. Zum Beispiel in Form von Musik oder einer Kirchenglocke.

Das Ziel ist, dass du lernst, was einen Ton genau ausmacht, indem du selbst verschiedene Klänge mit Gegenständen formen kannst.

Wie funktioniert das Experiment?



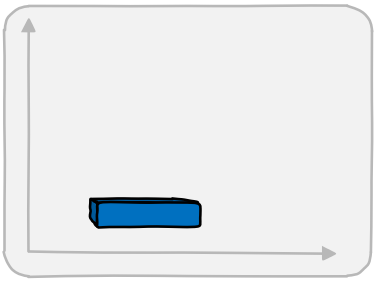
Auf dem Tisch sind zwei Richtungen eingezeichnet, welche den Bereich für dein Modell markieren.

Um einen Klang zu formen, nimmst du einen oder mehrere Gegenstände und ordnest sie auf dem Tisch an. In den folgenden Aufgaben sollt du herausfinden, wie sich der Ton verändern kann, wenn du die Gegenstände anders platzierst.

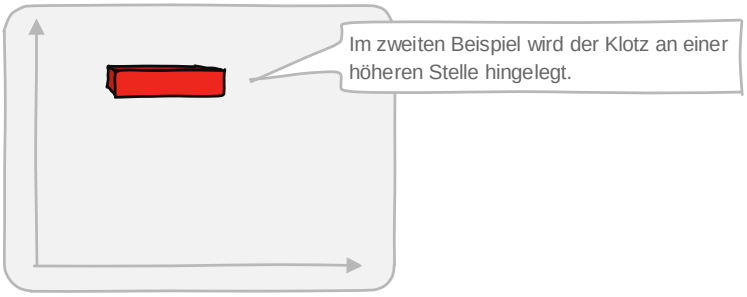
Sobald du den roten Knopf drückst, beginnt die Kamera dein Tonmodell einzulesen. Wenn die Berechnungen fertig sind, erscheint auf dem Bildschirm dein Modell und der Ton dazu wird abgespielt.

Aufgabe 1:

Nimm einen der Gegenstände und platziere ihn wie auf dem Bild im unteren Bereich. Drücke danach den Startknopf.



Verschiebe nun den Gegenstand gegen oben und drücke erneut auf den Startknopf.



Wie unterscheiden sich die Töne voneinander?

Aufgabenblatt: Modellierung von Tönen

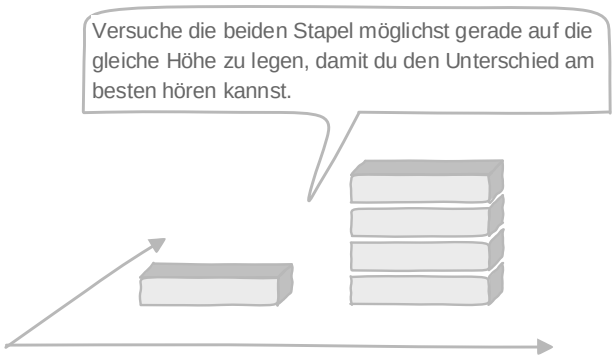
Aufgabe 2:

Versuche nun eine Form mit den Klötzen zu legen, die den Ton höher und tiefer werden lässt.

Zeichne die Form hier hinein

Aufgabe 3:

Bilde zwei Stapel unterschiedlicher Höhe und starte die Tonwiedergabe.



Was passiert mit dem höheren Turm?

Aufgabe 4:

Versuche nun die verschiedenen Eigenschaften eines Tones zu notieren:

Zusatzaufgabe:

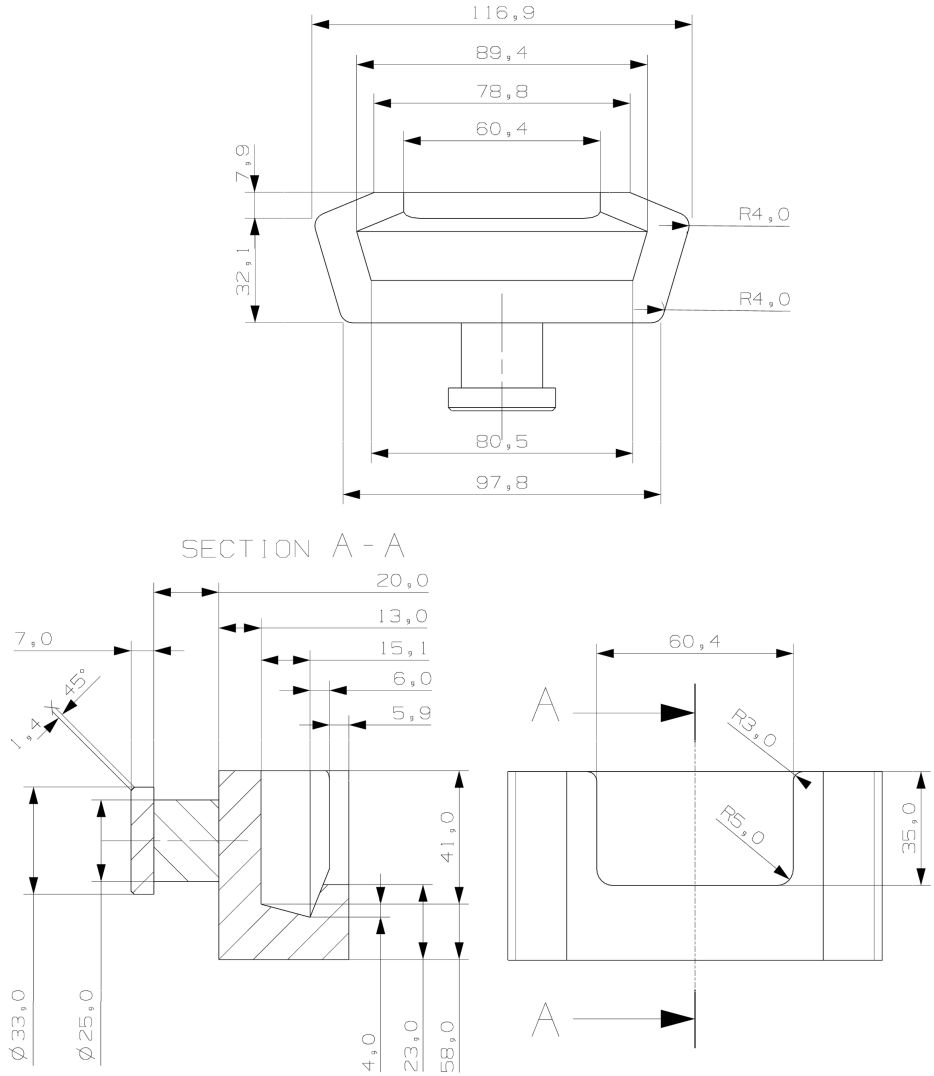
Zeichne eine Form ins Feld und bilde sie mit den Klötzen nach:

Wie wird sie sich anhören?

Appendix C

Datasheets

C.1 Camera mount



Bibliography

[1] M.R. Andersen, T. Jensen, P. Lisouski, A.K. Mortensen, M.K. Hansen, T. Gregersen and P. Ahrendt, "Kinect depth sensor evaluation for computer vision applications," tech. rep., Deepartment of Engineering, Aarhus University.

[2] Microsoft, "Kinect for windows - depth image artifacts." `http://msdn.microsoft.com/en-us/library/jj131032`. Accessed May, 2014.

[3] N. Burrus, "Rgbdemo." `http://labs.manctl.com/rgbdemo/`. Accessed May, 2014.

[4] C. Tomasi, R. Manduchi, "Bilateral Filtering for Gray and Color Images," in *Sixth International Conference on Computer Vision (ICCV)*, 1998.

[5] K. M. Wolter, *Introduction to Variance Estimation.* Springer, 2003.

[6] Edited by D. Purves, G. Augustine, D. Fitzpatrick, L. C Katz, Anthony-Samuel LaMantia, James O McNamara, and S Mark Williams, *Neuroscience. 2nd edition.* Sinauer Associates, Inc., 2001.

[7] D. F. Aldridge, "Mathematics of linear sweeps," *Canadian journal of exploration geophysics*, 1992.