A MODULAR COMPUTER VISION SONIFICATION MODEL FOR THE VISUALLY IMPAIRED

Michael Banf & Volker Blanz

Universität Siegen,
Media Systems Group, Germany
{banf, blanz}@informatik.uni-siegen.de

ABSTRACT

This paper presents a Modular Computer Vision Sonification Model which is a general framework for acquisition, exploration and sonification of visual information to support visually impaired people. The model exploits techniques from Computer Vision and aims to convey as much information as possible about the image to the user, including color, edges and what we refer to as Orientation maps and Micro-Textures. We deliberately focus on low level features to provide a very general image analysis tool. Our sonification approach relies on MIDI using "real-world" instead of synthetic instruments. The goal is to provide direct perceptual access to images or environments actively and in real time. Our system is already in use, at an experimental stage, at a local residential school, helping congenital blind children develop various cognitive abilities such as geometric understanding and spatial sense as well as offering an intuitive approach to colors and textures.

1. INTRODUCTION

Sonification is defined in [1] as “use of non-speech audio to convey information”. It describes both a scientific and an artistic discipline. In recent times it is even used to audibly browse the RNA structure [2] or model an acoustical representation of the standard model of particle physics [3]. In this paper we address the sub-field of developing sonification methods to support visually impaired people. In the last years, we have seen a number of special purpose devices that help visually impaired people to solve everyday problems, such as finding their way [4] or reading text. These devices tend to be restricted to a specific problem by extracting very specific information from the input data or by relying on additional information, such as positioning systems. In contrast, our goal is to develop a general-purpose system that can be used in a wide range of situations. Our system analyzes visual data using general image descriptors from computer vision, and maps these to auditory signals. Unlike approaches that use, for example, face detectors, we deliberately focus on low-level descriptors such as colors and edges in order to obtain a device that can sonify any given image and help to solve any given question that persons with normal vision could solve: We want to enable users to recognize what kind of scene is shown, find objects in this scene, or explore the photos of a friend. Unlike high-level image analysis, our device gives direct feedback on what is where in the image.

We are inspired by the way how visually impaired persons can explore reliefs of scenes with their fingers and get a direct haptic experience of the shapes and locations of objects – they can feel what the peak of a mountain or a what cloudy sky is. Due to the simplicity and directness of the sensory mapping from visual to auditory, we harness the human ability to learn, so we consider

Figure 1: Several exploring interfaces: (a) Pen to tablet interface with absolute coordinates. (b) Touch screen working directly on the acquired image. (c) Finger position (x,y) is tracked using a marker system and estimated within the image the brain of the user as part of the system.

There has been done previous work on the sonification of low-level characteristics of images for visual impaired. [5] generates sounds depending on a pixel’s lightness and its position within the image. [6] demonstrates a technique, using color patches within images as chromatic patterns that can be put together to form a melody. [7] uses color attributes to filter an underlying white noise using Subtractive Synthesis. [8] mixes 8 pre-recorded musical timbres depending on the quantity of 8 hue values within an image. All such synthesized sounds are not easily interpreted at first. Especially the last two approaches map all attributes of a certain color onto a single sound parameter, such as timbre, pitch or frequency spectrum. Such mappings are hardly reversible and might lead to identical sound synthesis from different input color combinations. Our contribution to previous work is to present a new and holistic approach to color and texture sonification, which is intuitive enough not only to be understood and applied by congenital blind, but also helps to convey e.g. the concept of colors and color mixing itself. We consider our approach to be holistic, as it maps each attribute of a particular color within a color-space – such as hue, saturation and lightness – to an intuitive, but separate, counterpart within the sound space at once. Additionally we map texture features to unique sound in the same way. The sonification of colors is important for two reasons, first, to offer a way to congenital blind people to understand colors and to be able to communicate with non-visually impaired about such fundamental quality of human vision. Second, in illuminated environments, colors are crucial in the process of detecting objects. If our sonification is intuitive enough, a person will quickly learn to understand the concept of colors and textures as well as the audition, recognize objects, interpret images and develop their own strategies. Designing such a system, we face the following challenges:

- Sensory input: We propose a system that analyzes images that users may find in a photo collection, on the internet, or capture with a still camera. Future devices could allow for depth information (stereo or time-of-flight devices), motion or other input.
Fels ar war, several tasks about what features shall be calculated and sonified are determined and passed through during exploration. Next step in the process is the creation of a pixel descriptor \((x, y)\) -- a vector gathering all the information to be sonified. The descriptor captures image information in the local neighborhood of \((x, y)\) and is calculated using computer vision techniques. This information might be very fundamental such as color, texture, edges as well as more complex such as an estimation of depth or whether the current pixel belongs to a face. Every feature makes an element \(d_i(x, y)\) of the pixel descriptor \(d(x, y)\) and is transferred to the sonification unit. In the next sections, we discuss each module and describe our specific design and its implementation.

3. SENSORICS

Our paper is on the interface between computer vision and sonification. Our contribution is the general concept of a Modular Computer Vision Sonification Model with the components Sensorics – Exploration – Machine Vision – Sonification – Human Learning, and a specific setup that implements this concept and that we present and evaluate below.

2. THE MODULAR SONIFICATION MODEL

Figure 2 gives an overview of the general concept of our Modular Computer Vision Sonification Model. Stage one is the acquisition of both a rasterized image \(I\) as well as a particular pixel position \((x, y)\) where the image is to be explored. Further, several tasks about what features shall be calculated and sonified are determined and passed through during exploration. Next step in the process is the creation of a pixel descriptor \(d(x, y)\) -- a vector gathering all the information to be sonified. The descriptor captures image information in the local neighborhood of \((x, y)\) and is calculated using computer vision techniques. This information might be very fundamental such as color, texture, edges as well as more complex such as an estimation of depth or whether the current pixel belongs to a face. Every feature makes an element \(d_i(x, y)\) of the pixel descriptor \(d(x, y)\) and is transferred to the sonification unit. In the next sections, we discuss each module and describe our specific design and its implementation.

4. EXPLORATION

Our exploration module has two major tasks. First, during the image acquisition, it allows the user to transmit control commands to the sensory module such as activating certain sensors or controlling their viewing direction. Second, it enables the user to navigate within an image, passing on its position \((x, y)\) to the descriptor computation unit, along with several tasks that determine which features of \(d(x, y)\) shall be sonified at all.

4.1. Navigation

Navigating within an image requires an appropriate interface. The computer-mouse, which is popular among users with normal vision, drops out as it does not deliver any absolute coordinates, which are necessary for a blind user to know the position in the image. Hence, we worked with several interfaces, as shown in Fig. 1, to see what suits best to a blind person. The pen – tablet interaction method functioned far better than the mouse, as it can be set to absolute coordinates. However, it turned out, that a direct touch helps to orient within the flat image, as analogous to moving the tip of the finger along a relief. Touch pads without a pen usually provide only relative positioning (similar to a mouse). For training and several user studies in section 7 we utilized a touch screen that allows the user to interact direct with the image plane (without seeing the image). Even though it does not make sense for a
blind person to buy a touch screen, the absolute positioning proved very successful. Therefore we implemented a more effective and far cheaper interaction method, applying a camera-based finger position tracking based on ARToolKitPlus pose tracking system [11]. The same camera as to acquire the visual data is utilized to detect a marker, attached to the user’s fingernail, which is thereafter calculated back to estimate the fingers position within the image.

4.2. Control Gestures

The use of a pose tracking system turns out to be of great interest in controlling the whole system as well. As the utilized ARToolKitPlus is able to deal with several markers at once it allows us to recognize many finger and hand gestures to submit control commands to the sensory module as well as sonification tasks to the descriptor computation unit. In contrast, a keyboard or virtual buttons on the touch screen are difficult to operate for blind users.

5. PIXEL DESCRIPTOR COMPUTATION

The pixel descriptor $d(x,y)$ holds all relevant features to be sonified at an image position $(x,y)$. We focus on fundamental characteristics such as colors and what we call Orientation maps and Micro-Textures. However, as intended, the module can be extended to extract and store more complex information.

5.1. Color Information

There are different color systems with several motivational backgrounds, as shown in Fig. 3. The RGB model uses additive mixtures of red, green and blue. It is motivated by the human eye receptors [12] and applied, e.g. in many display devices. However, providing a non-visual access to colors, as in our case, requires a more intuitive system, especially for congenital blind persons. This is why we prefer the HSL model [13], where each color value is described by hue $h$, saturation $s$ and lightness $l$. What makes color sonification difficult is the fact that color values often change rapidly from pixel to pixel even if there are only minute variations in textures and materials. Often, the reason is image noise by the camera. It is obvious that such changes clearly overburden a blind user. Therefore we smooth the image patch around the pixel position $(x,y)$ based on Bilateral Filtering [14], as shown in Fig. 4, which filters noise while preserving edges within an image and will play a significant role in finding orientation maps. Subsequently, we use the smoothed color values as the first three elements of our pixel descriptor $d(x,y)$:

$$
d_1(x,y) = h_{\text{smooth}}(x,y), \quad d_2(x,y) = s_{\text{smooth}}(x,y), \quad d_3(x,y) = l_{\text{smooth}}(x,y)$$

5.2. Orientation Maps

The rationale behind Orientation Maps and Micro-Textures is to create something like an acoustical relief that allows the user to hear what is under his fingers, instead of feeling it. While micro-textures - explained in the next section - express the overall roughness characteristics of a particular patch, orientation maps represent dominant structures within the image. We consider dominant structures single or repetitive sets of significant edges of the same orientation and a particular direction of propagation. Our method is based on the observation that standard edge detectors such as Canny [15] produce multiple edges and spurious, misleading signals that confuse the user. Therefore the calculation of orientation maps involves filtering important from distracting structures, which may be motivated biologically from the Surround Inhibition in the human visual system that improves contour detection [12]. Moreover, regular repeating patterns of a certain size tend to be human made, unlike the more fractal patterns that are often found in nature [16]. Finding human made structures is important when using the system to orient within an environment to find windows, doors, ways, tables, shelves and so forth.

5.2.1. Calculating Orientation Maps

In the first step of calculating orientation maps, we use cascades of Median [17] and Bilateral Filtering to suppress both noise and small corners, as shown in Fig. 4 (b). Then, we reduce the spatial resolution of the red, green and blue color channels to obtain a Gaussian Image Pyramid [18], as shown in Fig. 5 (a). On each level, the width and height is reduced by a factor of 2. This reduction process removes low-scale variations which may be irrelevant for the task of the user. By later combining information of different layers using the image pyramid we can select the most appropriate resolution for each visual feature. Next, we perform a Gabor Transform [19] on each channel separately and build a final response image by measuring all individual responses. The transform relies on Gabor Wavelets $\psi_{\nu,\varphi}(x,y)$ [20] of the form:

$$
\psi_{\nu,\varphi}(x,y) = \frac{k_{\nu,\varphi}}{\sigma^2} \left[ e^{-\frac{k_{\nu,\varphi}^2}{2 \sigma^2}} e^{-\frac{(x-x_0)^2}{2 \sigma^2}} \right] e^{i \nu \cdot x + \varphi}
$$

where the parameters $\nu$ and $\varphi$ define the orientation and scale of the Gabor kernel and $\sigma$ is the standard deviation of the Gaussian window in the kernel, i.e. the size of the window. $k_{\nu,\varphi}$ is the wave vector, combining orientations and the spatial frequency in the frequency domain. Gabor Wavelets are widely used in computer vision [20], because they provide an analysis of spatial frequency that is local, unlike the global analysis in a Fourier Transform [17]. To filter orientation maps, we choose $\nu = 1$ and later combine their particular response – inspired by the cascading of several simple cells to form Complex Cells [12] in the human visual cortex. We apply them in 32 orientations $\varphi = \{0^\circ, 5.625^\circ, 11.25^\circ, 16.875^\circ, \ldots, 180^\circ\}$. For the carrot and the orange in Fig. 4 (a), such response images are shown in Fig. 5 (b). The 32 orientations $\varphi$ are visualized by gray scale values. We
quantize the response image so that the each of the 32 orientations \( \theta \) is mapped to the next of 8 orientations \( \theta = \{ 0^\circ, 22.5^\circ, 45^\circ, 67.5^\circ, 90^\circ, 112.5^\circ, 135^\circ, 157.5^\circ \} \). Each \( \theta \) is represented in an individual gray scale image \( G_\theta \), where \( G_\theta(x,y) = 255 \), in case of an edge and \( G_\theta(x,y) = 0 \) otherwise. One fundamental idea of orientation maps is that the user should not have to follow contours of objects or structures to estimate their silhouette, which would be tedious and slow. To make it easier for users to find contours, we distribute them around the edge by a kind of diffusion approach. Therefore, we calculate for each image position \( (x,y) \) the variance \( \sigma(x,y) \) of values \( G_\theta \) on its local neighborhood on each of the 8 images \( G_\theta \), to obtain what we call Orientation Maps \( O_\theta \):

\[
O_\theta(x,y) = \sigma^2(x,y) = \sum_{i=-w/2}^{w/2} \sum_{j=-w/2}^{w/2} (G_\theta(x+i,y+j) - \mu(x,y))^2
\]

with

\[
\mu(x,y) = \frac{1}{w^2} \sum_{i=-w/2}^{w/2} \sum_{j=-w/2}^{w/2} G_\theta(x+i,y+j)
\]

where \( \mu(x,y) \) is the mean and \( w \) the size of the local neighborhood. Orientation Maps \( O_\theta \) for the carrot and orange (Fig. 4 (a)) are shown in Fig. 6. Each coherent patch of gray scale pixels, on each orientation map is referred to as Orientation Patch \( V_{\theta} \). As the diffusion approach might cause overlap in image positions \( (x,y) \) of different oriented orientation patches, as illustrated in Fig. 7 (a), we now compare orientation patches and emphasize the dominating ones while suppressing insignificant and therefore distracting others.

5.2.2. A Topological Representation of Orientation Patches

We consider orientation patches as dominant if they have a certain size — which is the number of their pixel positions. So far, we handle 4 cases. Case 1: If an image area is dominated by two very big orientation patches of almost equal sizes, both above a certain threshold \( t_{\text{size}} \), such patches differ in orientation by more than \( 22.5^\circ \), they are retained as coexisting. This happens to be the case for a rectangular grid or a wall of bricks, where two orthogonal orientations are permanently present and form the particular textures of the image region. Case 2: If the image area is dominated by two orientation patches, both greater than \( t_{\text{size}} \), having an orientation difference of only \( 22.5^\circ \), which is the smallest possible difference, these patches are combined into a single orientation patch by merging the smaller one into the bigger one. Each pixel of the smaller orientation patch is assigned to the orientation map of the bigger one. After that, the smaller orientation patch is erased from its orientation map.

Case 3: If the image area contains a large orientation patch, with a size greater than \( t_{\text{size}} \), and further patches, whose sizes are below \( t_{\text{size}} \), such smaller patches are either (a) merged into the big one, as described in case 2, or (b) deleted from their orientation maps, depending on whether their particular centers lie within the large orientation patch or not. Case 4: If the image area contains several orientation patches, whose sizes are below \( t_{\text{size}} \), they are merged as in case 2, in case both their center positions overlap and their orientation difference is equal to \( 22.5^\circ \). Otherwise they coexist. To implement the rules described in this section, we build a topological representation of all overlapping orientation patches in that area by the following procedure. At first, we apply a contour finding algorithm [21] on each orientation map \( O_\theta \) that retrieves a sequence of all contour pixels as well as all enclosed image positions found within the map. Each contour, as well as the enclosed pixel positions, belong to an orientation patch \( V_{\theta} \). We now represent each \( V_{\theta} \) as a single image, as illustrated in Fig. 7 (a). So far, for each \( V_{\theta} \), we have its size and can calculate its center of gravity. Starting with a particular orientation patch \( V_{\theta_1} \), we now check pixel by pixel for overlaps with each different oriented orientation patch. In case of overlapping, we compute the number of mutual pixel positions and whether the center of one orientation patch lies within the other. The results can be processed by graph theory [22] in the following way. Overlapping orientation patches can be modeled as an undirected Graph \( G_{\theta} = (V,E) \), where knots \( V \) represent all orientation patches and edges \( E \) represent the existence of an overlap between a pair of different oriented orientation patches. A second graph \( G_{\theta} \) is set up to represent only such overlaps, where at least the center \( c \) of one of the two orientation patches involved is found inside the related orientation patch, as visualized in Fig. 7 (b). In this case connections may be only in one direction, so \( G_{\theta} \) is a directed graph. Both graphs \( G_{\theta} \) and \( G_{\theta} \) for all orientation patches of the carrot image (Fig. 6 (a)), are illustrated in Fig. 8. Such topological representations can now be used to find which of the 4 previously mentioned cases fit. For the carrot image we find, based on \( G_{\theta} \) and \( G_{\theta} \) (Fig. 8 (c)) that \( V_{\theta_1} \) and \( V_{\theta_2} \) merge into \( V_{\theta_1,\theta_2} \), applying case 3a and \( V_{\theta_3} \) is deleted, applying case 3b. Hence, the final orientation maps \( O_{\theta_1}, O_{\theta_2} \) and \( O_{\theta_3} \) are shown in Fig. 7 (c). In contrast, the Graph \( G_{\theta} \) based on the orientation maps of the orange image, Fig. 6 (b), would result in a ring shaped connected structure, where every knot would be about the same size. Hence, such orientation patches are left as they are, according to case 4. Eventually, we can assign the shares of all eight orientation maps at a particular pixel position \( (x,y) \) to the pixel descriptor...
We define $d(x,y)$ to be either 1, in case $O_0(x,y) > 0$, or 0, if $O_0(x,y) = 0$.

Note that as borders of orientation patches fade out, caused by micro-textures we use the roughness information. Unfortunately, a major drawback are high calculation costs.

5.3. Micro Textures

What we call Micro-Textures in our system captures the roughness and local structure at a point in an image. Examples of textures that we want to distinguish are shown in Fig. 9: regular lines, brick walls, in-grain wallpaper or carpet. In the Computer vision literature, there are many approaches to describe texture, even though it is difficult to give a general definition. [13] describes three different groups of texture measures. First, there are First Order Statistical Texture Measures [17] such as the mean $\mu$ and variance $\sigma^2$ of color values in a local neighborhood. Second, there are Second Order Statistical Texture Measures. These texture measures do not analyze single intensities, but correlations between pairs of pixel values. An example of this approach are Haralick texture measures based on Co-Occurrence-Matrices [23]. Unfortunately, a major drawback are high calculation costs. There have been efforts to speed up the processing using GPU [24]. Finally there is Spectral Image Analysis such as e.g. the Fast Fourier Transform [17] or Gabor Transform. To build micro-textures we use the Gabor Transform and compute the Entropy as a texture measure.

5.3.1. Entropy as Texture Measure

The Gabor Transform is applied to the original image and not to the bilateral filtered image, as we now want to preserve roughness information. Entropy, generally measures the disorder within a physical system, and is used in various scientific fields. Having zero entropy means to have maximum information about the state of a system [25]. In information theory, it is formulated as:

$$H = - \sum_{i=1}^{N} p_i \log p_i \quad \text{with} \quad p_i = \frac{N_i}{N} \quad (3)$$

We calculate the Entropy $H(x,y)$ for each pixel position $(x,y)$, based on the Gabor Transform response within a local neighborhood. The variable $p_i$ is the probability for a certain orientation $\varphi_i$ estimated from its occurrence $N_i$ divided by the total number $N$ of all orientations $\varphi$ that occur within the window. Based on $H(x,y)$ we can now measure the roughness of an image region and assign it to the last element of $d(x,y)$:

$$d(x,y) = \begin{cases} 0 & \text{smooth surface, if } N < N_{\text{max}} \\ 1 & \text{anisotropic roughness, if } 0 \leq H(x,y) < H_{\text{intrigene}} \\ 2 & \text{isotropic roughness, if } H_{\text{intrigene}} \leq H(x,y) \end{cases}$$

Figure 9: 4 types of textures with increasing Entropy.

5.4. Grabbing Objects

We utilize state of the art segmentation algorithms to separate the image into regions that are likely to show different real life objects. The goal is to help users find and scrutinize objects and other entities in the image, based on orientation maps and micro-textures calculated for such parts only. Image Segmentation or more precisely a foreground / background segmentation - is performed using Gaussian Mixture Models calculated using Expectation-Maximization and Graph Cuts [26, 27]. First, the user moves an area of interest and initiates the segmentation procedure by pressing a button or by gesture. Based on the users current position $(x,y)$ we apply a flood-fill algorithm [29] that iteratively adds pixels to an area around $(x,y)$, if their color distance to the average color of the region is below a threshold. The color distances are calculated as the Euclidean distance $[\ldots]$ in Cielab [13] Color Space, as illustrated in Fig. 4 (c). All these selected pixels are marked as "definite foreground", all others as "probably background", and both groups serve as first segmentation estimation and as input to the Expectation Maximization and Graph Cut algorithms, which then calculate the final segmentation. Fig. 10 shows some exemplary results of the segmentation, as well as the responses of the Gabor Transform applied to such segmentations. The whole segmentation process takes approximately 4.5 seconds and can therefore be considered for interactive usage.

6. Sonification Concept

The previous sections dealt with extracting features to form the pixel descriptor $d(x,y)$. We now discuss the question of how to sonify those features. A great challenge is to avoid conflicting signals and information overload, as well as the transformation of quasi-static 2D image data into a dynamic audio stream. Though humans can distinguish many attributes such as pitch, volume, ADSR-Curve, timbre, roughness or vibrato, it is still impossible to transport all potential descriptors of visual information simultaneously. Unlike approaches that sonify a whole image sequentially e.g. by scanning its pixels row by row [30] we want the user, as already described, to fully interact with the visual data in real-time and to be able to hear what is currently under his finger. Second, we want a method to simultaneously sonify features such as color, orientation maps and micro-textures and even more, instead of focusing on a single feature such as the progression of edges [31], [32]. Third the sonification model should meet aesthetical demands that are important for comfortable and extensive usage.

6.1. Sonification of Color Information

Sonification systems may use different techniques of sound synthesis, such as Subtractive Synthesis, Additive Synthesis, Granular Synthesis, Physical Modeling or FM Synthesis [33]. The methods presented in this paper rely on sounds from common instruments based on the General MIDI (GM) Standard (based on Wavetable Synthesis [33]). Visual impaired people
may find this a comfortable and soon a familiar way to get perceptual access to colors and textures. Instead of strictly learning particular associations between instruments and colors (which they do not see and therefore have to memorize) we want to help them to build connections between sonification signals and objects of their daily life. In fact, in our experiments, we often heard participants say that an image region "sounds like a tomato" or any other object rather than announcing the correct mixture of colors, such as red – as in case of the tomato.

6.1.1. Complementary Instruments

As we use common instruments to sonify visual information, we propose a concept that represents each color value in a mixture. This criterion does not need to be met by mutual masking of instruments, their frequency spectra should represent the luminance scale from black to white. As no mixture of a pair of opponent colors exists [12], there will be no mixture of a pair of complementary instruments in the sonification model either. Further we apply a musical scale to represent the color mixture ratio by their partial volume. The fade of saturation \( s \), moving inward to the center of Fig. 11 (a), is considered as a general absolute decrease in volumes of any two color instruments playing simultaneously, while their relative volume ration is maintained. However, below a certain threshold \( s_{\text{th}} \), we regard the color as gray and sonify it using a single instrument, the flute. In general, gray is not considered a color, and the HSL model assigns it an arbitrary hue \( h = -1 \) and a saturation \( s = 0 \). Still, we found it helpful to use a separate instrument for gray, which partly reflects the fact that many languages have a separate name for it. The lightness \( l \) of gray or any other (combination of) colors is sonified as the pitch of the tone. Gray scale images, therefore will be sonified as a flute playing at varying pitch. Based on a musical scale, as shown in Fig. 11 (c), black, as the lowest lightness value, is assigned to the tonic keynote, whereas white to its octave. In between there are six whole tones and eleven semitones. For harmonic reasons we only utilize the whole tones of the octave and map each lightness value \( l \) between 0 and 1 to one of the eight tones. Further, we add thirds to all six intermediate tones. This creates a more comforting and aesthetical resonance and offers an elegant way to recognize whether one has reached the top or bottom of the scale, as they are played without thirds. Otherwise, users would need perfect pitch to recognize black and white. When working with scales in MIDI, each note has to be triggered and released, which, again, is why a very short Attack- and Decay-and as well a short Release-Phase is essential to maintain a close-to-continuous signal. In contrast, mixing colors on a constant luminance takes place solely within the Sustain phase for arbitrary time – the note itself does not change.

6.1.3. Calculation of Volume Shapes

Calculating the volumes of instruments in a mixture of sounds for all intermediate colors is an interpolation problem. Simple linear (barycentric) interpolation would be too restricted because once the overall volume of each instrument is set, there would be no way to counteract the dominance of some instruments in some specific mixtures. Therefore, we use thin plate spline interpolation [34] based on a set of control points. The fundamental idea behind the method is the physical model of a flat thin medal plate as in Fig. 12 (a) that is deformed by a few punctual strains, which we will call control values \( c \). The plate is then forced into a new form that minimizes the
Proceedings of the 18th International Conference on Auditory Display, Atlanta, GA, USA, June 18-21, 2012

We decided to utilize four more instruments, playing an octave lower than the keynote, to represent orientation maps of different motivations. First, as a proof of concept of our framework, we asked a congenital blind, 54 year old adult, who had acquired a geometric understanding and spatial sense throughout his life to solve several tests after 4 hours of training with our system. The participant was to solve three naming tasks at increasing difficulty:

- **Experiment 1a** was about identifying one out of four elements (orange, tomato, apple and lemon — as in Fig. 13 (a)) only by color while sonification of orientation maps and micro-textures was deactivated. Note that the target objects used for the task have the same spherical shape. In each of 60 trials, one of the 4 objects was selected at random and displayed at an arbitrary position on the touch screen. This was achieved by selecting one out of 40 images (10 per object, with the object in different positions) at random. The task of the participant was to find and name the object. In the evaluation, we focus on the time between the moment when the participant finds the object (which depends on where he starts and is therefore not very informative), and the moment when he names the object verbally to the experimenter (Table 1 and Fig. 14). The average time to simply find an object’s position on the screen was about 1.7 seconds. Chance level (pure guessing) is 25% in this experiment.

- **Experiment 2a** involved orientation maps and color. This time, the participant had to recognize one out 7 objects (orange, tomato, apple, banana, cucumber, carrot, lemon), as shown in Fig. 13 (a), so both color and shape are important for correctly naming the object. Again, each element was presented individually (chance level: 14%) at arbitrary positions and also in one of eight orientations, as illustrated in Fig. 13 (b). The database consisted of 56 images (8 for each element, varying position and orientation). Again, times were measured between finding and naming the object verbally, as shown in Fig 14 and Table 1.

- **Experiment 3** was about recognizing an object within a set of other objects. Therefore, we presented images like the

---

**Figure 13:** (a) Test set. (b) Possible orientations of objects.  
**Figure 14:** Histograms for (a) Exp. 1a/b and (b) Exp. 2a/b. N elements (y-axis) recognized in how many sec. (x-axis) each.
one shown in Fig. 13 (a) on the touch-screen. In our database of 7 images, we made sure that two objects of equal color (e.g. banana and lemon) would not be positioned next to each other. In each trial, an image was presented and the participant was told, which object he had to find, based on a random generator. This time, we measured the overall time until an element was named.

In Experiment 1b and 2b, we tested the system on a group of congenital blind 14 year old teenagers. Unlike the adult participant, they had little geometric understanding and sense of space. We hope that our system can not only support them in everyday life, but also help them to develop cognitive abilities in geometry and spatial orientation. We performed an experimental evaluation of our system to measure their progress and compare it with the results of the adult participant. Three teenagers were trained about 5 hours with the system. This also included fundamental lecturing about basic geometry. Then, we asked them to perform Experiment 1b and 2b, which had the same setup as 1a and 1b described above. Surprisingly, the teenagers were able to perform the tests with similar hit rates and times as our adult participant (Table 1 and Fig. 14). We have presented a general framework and a sample implementation of a device that can support blind and visually impaired persons in exploring images or scenes. Many details of our implementation, such as the choice of local image descriptors, may be modified or improved further. However, we tried our best to design descriptors that are most promising from the theoretical and most informative from the practical point of view. The same is true for our sonification concepts: they are only one way how this can be achieved, yet we argue that it is an appropriate and powerful way to do it.

We thank the Rheinischen Blindenförsorgeverein, Dürren, especially Marina, Larissa, Florian, Sascha and Mrs. Gut for their interest and participation in the project. We also thank Tobi and Rainer for their highly appreciated advisory support.

9. ACKNOWLEDGEMENTS

10. REFERENCES


Our vision is to provide visually impaired persons with software for web browsers, image "viewers", and on portable systems such as smart phones.

8. CONCLUSIONS

We thank the Rheinischen Blindenförsorgeverein, Dürren, especially Marina, Larissa, Florian, Sascha and Mrs. Gut for their interest and participation in the project. We also thank Tobi and Rainer for their highly appreciated advisory support.

<table>
<thead>
<tr>
<th>RESULTS</th>
<th>P</th>
<th>Hit rate</th>
<th>X</th>
<th>( \mu )</th>
<th>( \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp. 1a</td>
<td>A</td>
<td>100.0</td>
<td>60/60</td>
<td>1.3</td>
<td>1.8</td>
</tr>
<tr>
<td>T</td>
<td>91.6</td>
<td>55/60</td>
<td>2.2</td>
<td>2.6</td>
<td>1.5</td>
</tr>
<tr>
<td>Exp. 1b</td>
<td></td>
<td>93.3</td>
<td>56/60</td>
<td>1.5</td>
<td>2.3</td>
</tr>
<tr>
<td>T</td>
<td>100.0</td>
<td>60/60</td>
<td>1.3</td>
<td>1.7</td>
<td>1.1</td>
</tr>
<tr>
<td>Exp. 2a</td>
<td>A</td>
<td>93.3</td>
<td>42/45</td>
<td>5.6</td>
<td>7.0</td>
</tr>
<tr>
<td>T</td>
<td>88.8</td>
<td>40/45</td>
<td>12.1</td>
<td>13.3</td>
<td>4.7</td>
</tr>
<tr>
<td>Exp. 2b</td>
<td></td>
<td>93.3</td>
<td>42/45</td>
<td>11.9</td>
<td>12.5</td>
</tr>
<tr>
<td>T</td>
<td>88.8</td>
<td>40/45</td>
<td>10.1</td>
<td>11.4</td>
<td>6.6</td>
</tr>
<tr>
<td>Exp. 3</td>
<td>A</td>
<td>100.0</td>
<td>45/45</td>
<td>5.6</td>
<td>10.6</td>
</tr>
</tbody>
</table>

Table 1. Hit rates and times (median \( X \), mean \( \mu \), and standard deviation \( \sigma \), for each trial and participant \( P \).