Hannu Seppälä

3 Applications for Visually Impaired

Metropolia University of Applied Science
Master of Technology
Information Technology, Master's Program
Thesis
8.8.2017

ABSTRACT

Author	HannuSeppälä
Title	3 Applications for Visually Impaired
Number of Pages	55 sivua
Date	7.8.2017
Degree Programme	Information Technology, Master's Program
Instructors	Juha Sylberg, Project Manager Harri Airaksinen, Principal Lecturer

This thesis describes an attempt to create 3 applications for smartphone to aid visually impaired. The applications should be able to guide a blind person in our modern environment.

The purpose of this study was to see if object recognition algorithms and supplemental algorithms can help visually impaired. The applications should be able to tell to the user useful information of the environment, such as distances to obstacles. Traffic signs are also objects to be recognized by the applications.

There are five basic ideas to begin the development. The first idea is to use (laser) distance measuring to obtain accurate distances to obstacles. The second idea is to use the camera to collect information about the environment. The third idea is to use direction to guide. The fourth idea is to design the UI in a way that a visually impaired person can use it. Finally, the idea of the project was to develop the applications together with the visually impaired.

Three working applications were developed. These applications do not work correctly in all situations. In such situations the applications should warn the user. This software development will be done in the future.

Keywords	Guidance, visually impaired, NDK, OpenCV

INSINÖÖRITYÖN TIIVISTELMÄ

Kirjoittaja Otsikko Sivuja Päivämäärä	Hannu Seppälä 3 Applications for Visually Impaired 55 sivua 7.8.2017
Koulutusohjelma	Information Technology, Master's Program
Ohjaajat	Juha Sylberg, Project Manager Harri Airaksinen, Principal Lecturer

Tämä työ kuvaa yritystä luoda 3 sovellusta älypuhelimeen näkövammaisten avuksi. Sovellusten tulisi kyetä opastamaan sokeaa modernissa ympäristössämme.

Työn tarkoitus on selvittää, voiko esineentunnistusalgoritmeja ja muita algoritmeja käyttää korvaavina silminä näkövammaisille. Sovellusten tulisi kertoa hyödyllistä tietoa ympäristöstä, kuten etäisyyksiä esteisiin. Sovellusten tulisi myös tunnistaa liikennemerkkejä.

Kehittämisen aloittamiseen on viisi ideaa. Ensimmäinen idea on käyttää etäisyysmittaria mitattaessa etäisyyttä esteisiin. Toinen idea on käyttää kameraa hyödyllisen tiedon keräämisessä. Kolmas idea on käyttää suuntaa opastuksessa. Neljäs idea on suunnitella näkövammaiselle sopiva käyttöliittymä. Lopuksi, ajatus oli kehittää sovellukset yhdessä näkövammaisten kanssa.

Kolme toimivaa sovellusta kehitettiin. Kaikissa tilanteissa sovellukset eivät toimi. Näissä tilanteissa sovellusten tulisi varoittaa käyttäjää. Tämä sovelluskehitys tehdään tulevaisuudessa.

Avainsanat	Opastus, näkövammaiset, NDK, OpenCV

TABLE OF CONTENTS

Figures

Code Examples

Introduction

This thesis describes an attempt to create applications to guide a visually impaired person. The applications should run on a smartphone. The aim was to cooperate with the end users who should affect the development.

First, the current applications were studied. The emphasis was on applications which help a visually impaired to move around using the phone. The aim was to find new ways to guide. The reason for this is that there are many guidance applications. Some of these the visually impaired can use even if they are not meant just for them. There are also guidance applications meant just for visually impaired. The study reveals features of the applications which are useful and some less so.

There were iterations of the development cycle. This means that first a demo version of the application was produced, then the application was given for the users to test. The users told the author what to do next to make the application work better. This proved to be a challenge, because it was difficult to agree on meeting times which all can make.

One purpose of this project was to see whether it is preferable to have three different applications or just one big application which houses all the desired features. These kind of current applications were tested. Some users found it hard to navigate lengthy menus. For this project it was decided to keep the three applications separate to ease the users to find what he or she is looking for in the applications.

Obstacle avoidance will have a key role in this project. There are many ways to avoid obstacles. There are also many types of obstacles to avoid. To determine the distance to an obstacle means that the application should be able to measure distances. This can be achieved using the sensors of the phone. There are also gadgets to attach to the phone, which measure distances. Which method will be chosen depends on the user tests and which will be the main obstacles to detect.

One question to answer during this project is how the application informs the user about the obstacles. This has to do with the overall design of the user interface. Being a phone application, sound probably will be used.

In the conclusions section of this paper some of the answers of these questions will be reported. All three applications will be published at Google Play. They will work on Android phones.

Programming will be done mostly using Java programming language. Eclipse development kit will be used. These will be applied using good practices learned at Metropolia. Other programming languages, like C++ will most likely be used. There will be new code created, especially when measuring distances using only one camera. Mathematics for measuring distances using monocular camera has already been developed, like Kanade - Lucas, but there are not many applications of it using a mobile phone. New algorithms and programs are being written all the time around the world about using the camera. It will be interesting to see whether these can be applied in this project.

The initial idea for this project was born during a lecture at Metropolia about the sensors of a smartphone. We were talking about accelometer and magnetometer and how they could help in certain situations.

2 Project Fundamentals

This chapter describes the background of the project. Some facts about visually impaired are introduced. Then problems which are encountered by the visually impaired are discussed. Moving around with the help of a smartphone is discussed next. Light and cameras are discussed, because cameras of smartphones operate using light. Cameras are then compared with human vision. Existing applications, which visually impaired use, are described. Finally, computer vision and the way the guidance is approached in this project are discussed.

2.1 Starting Points

In 2011, there were approximately 50 000 visually impaired people in Finland according to a study made by National Institute for Health. Out of these 50 000, 8 400 were considered blind. This means that about 1.6 % of the population are visually impaired. Being visually impaired can cause many inconveniences. Finding things can be difficult and moving around is not easy either, to mention some. These difficulties can lead to further injuries. Coping with injuries can be more difficult than for a person with an adequate eyesight. To resolve these issues visually impaired have developed different aids. [11]

To move around is essential for us all. It is both physically and mentally healthy to move around. For visually impaired moving around is a challenge both in countryside and in an urban environment, but moving around has challenges for us all. We do get disoriented. Not knowing our current position can happen. In this situation knowing the direction can be essential. A compass can provide some help there. Compasses have existed for long and usually they have been used when traveling far. Compasses have been especially helpful at sea, where there are not many landmarks. Knowing your bearings can be helpful in any travel. It can help find things even indoors. In addition to using compass, finding landmarks is essential for a traveler. For example, missing a traffic sign in traffic can cause a traffic accident. Collision avoidance is also an important part of any travel. To be able to avoid collisions requires measures from travelers and from those who plan and build environments.

Smartphones have been under significant development under resent decades. The memory and computing power of a smartphone have developed from Nokia 9000 Communicator with 24 MHz processor and 8 MB of memory to 1 GHz CPU and 4 GB memory according to current manufacturers. These features are adequate for using the camera

of the phone to recognize things. Modern phones have some other sensors their predecessors lacked. For example, modern phones usually have a magnetometer and accelerometer. These modern features could be used to aid visually impaired and some applications have successfully been developed. Some of these are described later in this paper.

Meetings with visually impaired at Näkövammaisten Keskusliitto, at Iiris, in Helsinki, formed the foundation for the project. It was observed that phones could help localization, obstacle avoidance and object recognition. The purpose of this project was to explore the possibilities to further develop applications for visually impaired. After studying the current applications it became clear that some areas of using phone as an aid were more developed than others. Location based applications, like guidance applications, were numerous. Some collision avoidance applications were also there for visually impaired.

All applications using cameras are based on light and its properties. A camera can detect almost all the properties of light as human eye can. For a human, losing an ability to detect some property of light may mean losing vision. Using cameras this ability may be restored to an extent. Some cameras operate better than human eyes in some situations. Night vision is an example of this. Cameras of the phones usually detect light's intensity and color.

There are applications to aid visually impaired in many ways. There are for instance readers. These read texts for the user. Usually they read texts which are already in a format that a computer understands. The more advanced applications can read any text, even those on a newspaper. The user points the camera towards the text and the application starts reading the text it finds. Magnifiers enlarge digital texts. In many cases magnifiers work together with readers. Usually the readers cannot read all fonts. [30]

2.2 Light

To operate, cameras need light. Light has many sources. Light can be produced for example by a glowing object. Most of the light we see comes from the Sun. Sun's nuclear reactions, as Hydrogen turns into Helium, produces gamma photons and neutrinos. The energy of gamma photons travels to the surface of the Sun, where ultraviolet radiation, light and infrared radiation are produced.

Light can be described as waves or particles. This is called dual model. Light particles have no mass. Momentum p, energy E, wavelength λ and frequency f have the following relation:

$$p = h/\lambda$$ [13]

and

$E = hf$, where h is Planck's constant. [12]

Interference and diffraction are phenomena related to waves of light. Wavelengths and colors correspond according to the following chart:

Color	**Wavelengths (nm)**
Violet	380 - 450
Blue	450 - 490
Green	490 - 560
Yellow	560 - 590
Orange	590 - 630
Red	630 – 760 [14]

Many famous scientists contributed to the understanding of light. Isaac Newton noticed that different colors land on different places in a telescope. He developed a telescope based on mirrors to gain better view of celestial objects.[15] James Klerk Maxwell developed his equations of electromagnetic radiation. He was able to calculate the speed of light using his equations.[16] Albert Einstein developed laser and studied the behavior of a light beam.[17][27] All of these efforts led to development of cameras.

Luminous intensity is an important aspect when using cameras. Luminous intensity is the measure of power emitted by a light source. The SI unit for it is the candela. Humans experience luminous intensity as brightness. Using Luminous intensity and color in computer vision are discussed later in this paper.[18]

2.3 Cameras and Eyes, Seeing

Most cameras used silver halide, which turns dark under light, before digital cameras were developed. Digital cameras capture the incoming light rays and turn them into electrical signals. Light detectors are used to collect light. These light detectors are usually either a charge-coupled device (CCD) or a CMOS image sensor. CCD is based on semiconductors. Light causes electrons to move causing a current. The more light the more current. Usually there are three filters: Red, blue and green. CMOS is also a semiconductor. It has low power consumption and it uses a single power supply. Of the two CCD is the most used in cameras. It has smaller pixel size. [19]

In Figure 1 there is a CCD. Electron movement caused by light is transferred to the neighboring capacitor. This transfer is controlled by a separate unit. The last capacitor in the array sends the charge to an amplifier, which converts the charge to voltage. The result is a sequence of voltages which are sampled and digitized. This image was found at: https://www.unifore.net/analog-surveillance/security-camera-ccd-cmos-image-sensor.html.

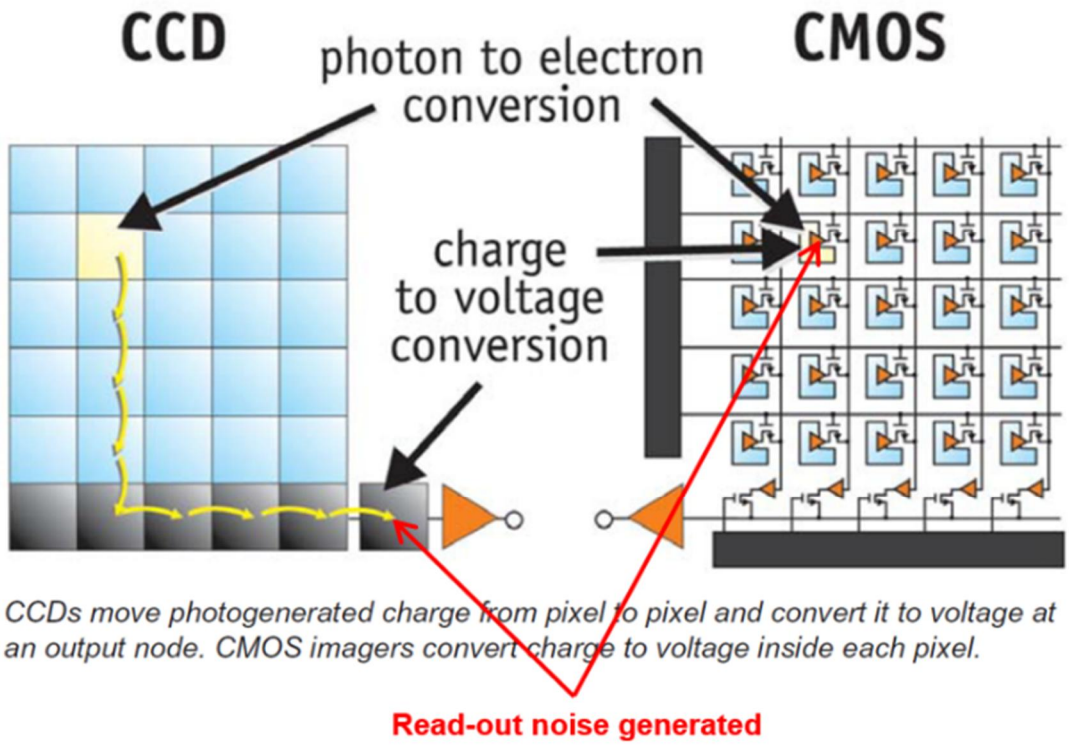

Figure 1. CCD and CMOS.

A light ray must hit the right area of the CCD or a human eye. A camera and an eye use a lens. In an eye it focuses light onto a light-sensitive membrane called the retina. The cornea is a transparent structure found in the very front of the eye that helps to focus incoming light. Situated behind the pupil is a colorless, transparent structure called the crystalline lens. A clear fluid called the aqueous humor fills the space between the cornea and the iris. There are muscles to stretch the lens. These muscles try to focus light correctly. Failing to do so can be a cause for a person to become visually impaired.

An eye and a camera focus light differently. A camera focuses all colors similarly. Light rays must reach the same spot for different color filters. An eye focuses colors at different places of retina. There are 4 kinds of cells on retina. Three kinds of cells are used during daylight. These receive the red, blue and green light rays. The fourth cell type is used when it is dark. Humans do not see colors when it is dark. Figure 2 provides an illustration of an eye with main operational parts.

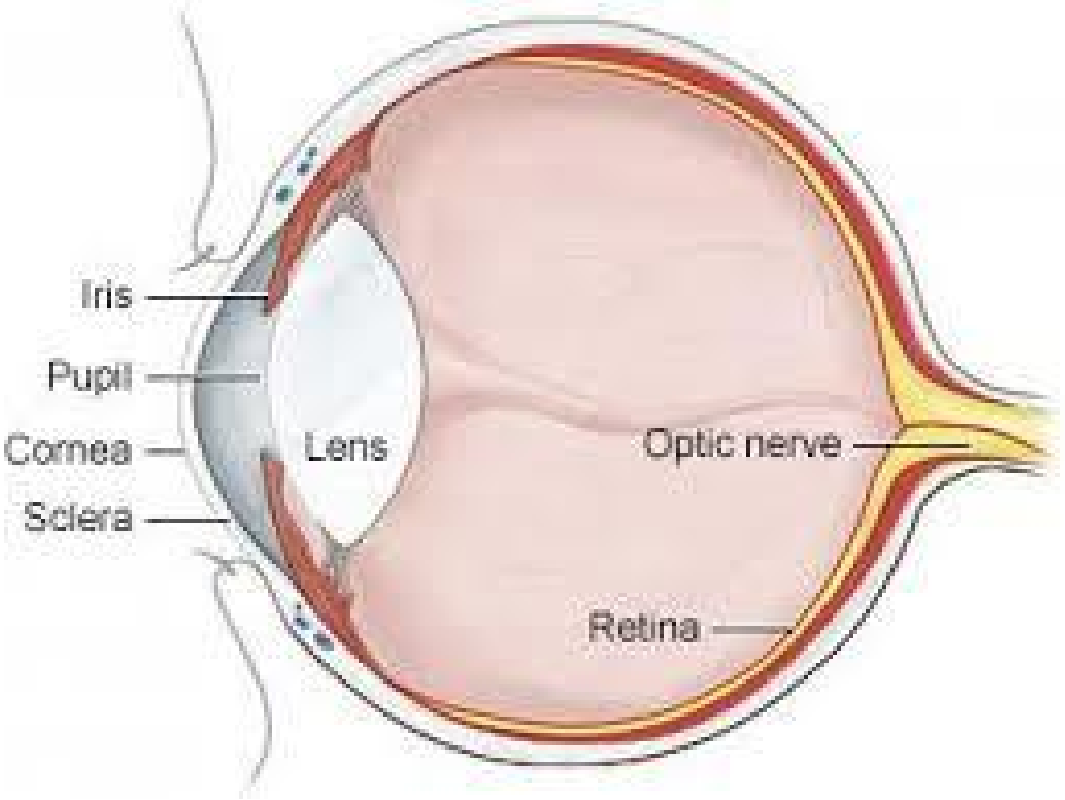

Figure 2. An Eye

An injury to the eye can have many causes. Diabetic retinopathy is the most common cause of injury to the eye in Finland. Diabetic retinopathy is vision-threatening damage to the retina of the eye caused by diabetes. Globally deficiency of vitamin A is also a thread to vision. Premature birth can also threaten development of vision. In recent years work related injuries to eyes have declined in Finland. [20]

Many insects have their eyes so close together that their sight can be considered monocular. Insects seem to use optical flow as means to maneuver in an environment. They may use angular velocity to estimate distances to objects. This could be something they do all the time for instance during flight. The closer the object is the greater the angular velocity in in relation to the insect. Monocular vision is discussed later in this paper. [26]

If seeing is defined as an ability to adequately gather information about environment to move around, losing vision totally does not mean that a person then cannot see. In that case there are still means to see. Seeing happens as a function of the brain. [21] [23] All other senses can be used to aid seeing. Without sight we use touch, sound and sometimes even smell to move around. In a totally dark room we can walk to the next wall and when we arrive there, we feel the wall. Using touch we can memorize where the obstacles are. Perhaps the most used senses in a situation where the sight is lost, are sound and touch. Losing vision does not mean losing the ability to move around. This became obvious while working with the visually impaired.

2.4 **Moving around and Locating**

Knowing which way to go means that a person must know where he or she is at. There are several ways to do this. One way to do this is to use the smartphone. Most smartphones have GPS, which provides a rough measure of the location.

To help the user to localize himself or herself several applications have been developed. Some of these are meant for visually impaired. Ilkka Pirttimaa has developed a GPS based application for the blind called BlindSquare (http://fi.blindsquare.com/about/). The application uses Foursquare API (https://developer.foursquare.com/) to alert the user about nearby places around user's location. Foursquare is a database containing information about places. BlindSquare works on iPhones and iPads. The application supports bone conducting earphones. These are used because normal earphones would

prevent a blind person from hearing surrounding sounds. The blind often use sounds to navigate. BlindSquare can guide a person to a selected place by telling the direction and the distance to the place. By shaking the phone the user is told the current address, nearest crossroads and current direction. FourSquare allows the user to add own places to FourSqare. This must be done with the help from a person who can see.

BlindSquare is under development. An interview with Ilkka Pirttimaa revealed that in the future the user perhaps will be told by his application from where the user is approaching the desired location. In other words the application calculates the direction and the distance the user must go in order to reach the desired location. There are a number of navigation applications for Android and iPhones. Some Android applications are described in the following paragraphs.

WalkyTalky(https://play.google.com/store/apps/details?id=com.googlecode.eyesfree.walkytalky&hl=fi) is one of many navigation applications for visually impaired. It was developed by Google (https://www.google.com/intl/en/about/). WalkyTalky reads the nearest street address aloud. It also speaks instructions from intersection to next. The routes it navigates are pedestrian routes. The user can save the favorite destinations. These destinations can be navigated to. Recent destinations are also saved by the application. It also provides useful information about the destination. It can for example tell whether the destined shop is open or closed. If the user walks the wrong way, the phone vibrates. A useful way to navigate using android is using many navigation applications simultaneously. The basic idea of WalkyTalky is to use Google Map Navigation in Walking Direction mode. These are launched directly at the start of the application. [10]

In Figure 3 the location of the user's location has been put on the map by the application. The user has told that he or she wants to go to Kampin keskus. The application then navigates the user there. The application guides the user along pathways and pedestrian crossings.

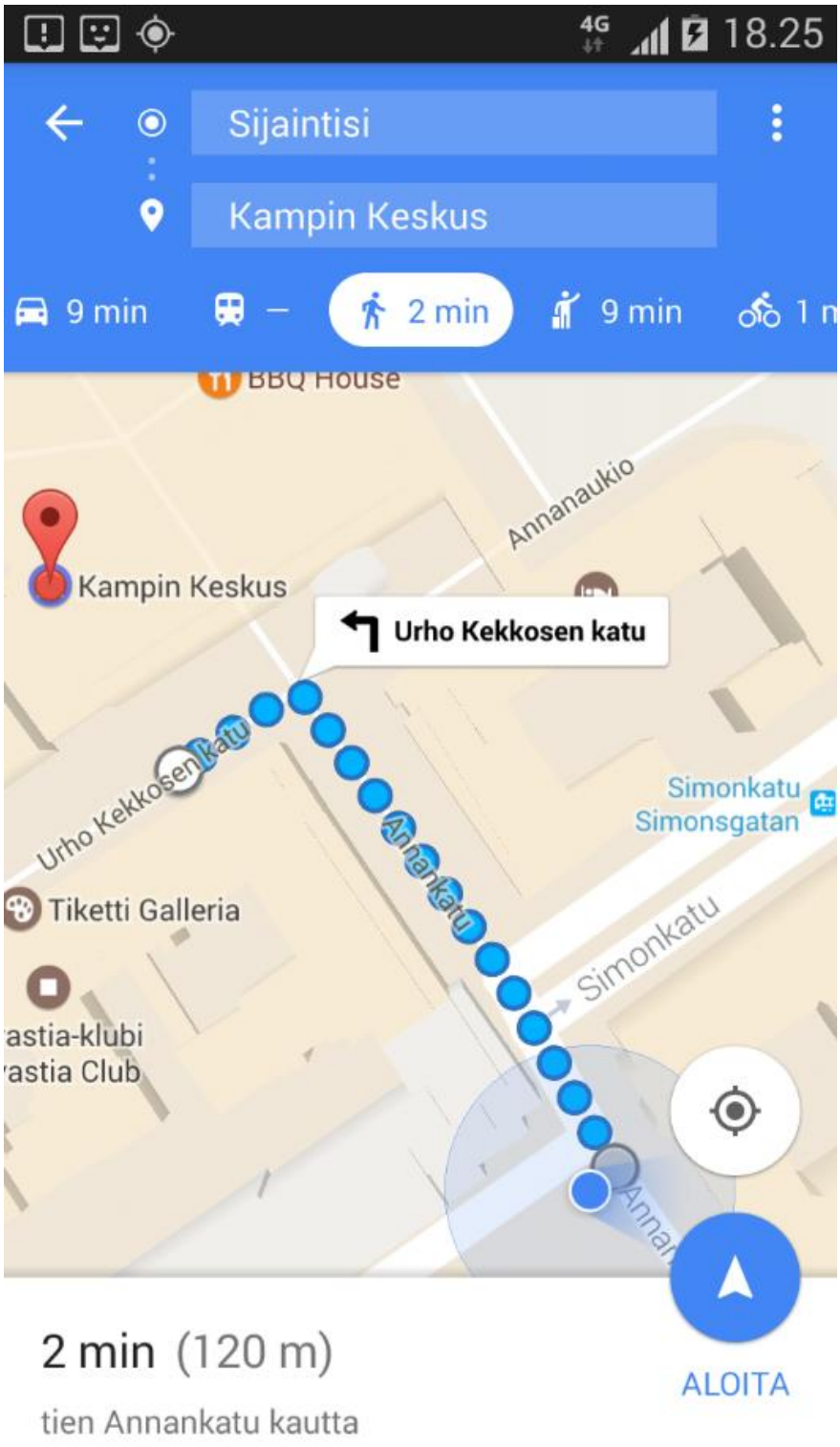

Figure 3. WalkyTalky in action

IntersectionExplorer(https://play.google.com/store/apps/details?id=com.google.android.marvin.intersectionexplorer&hl=fi) allows the user to explore the near

neighborhood by moving finger on the screen of the phone. The street names of the intersection nearest to the finger press on the map and the distances are spoken. This application could be useful in a familiar environment (Where the names of the streets mean something for the user.). This application was also developed by Google. Figure 4 shows a small red spot where the user touched the screen. The user has been located and the user has pressed the center of the screen. The text at the top of the screen is read out loud by the application.

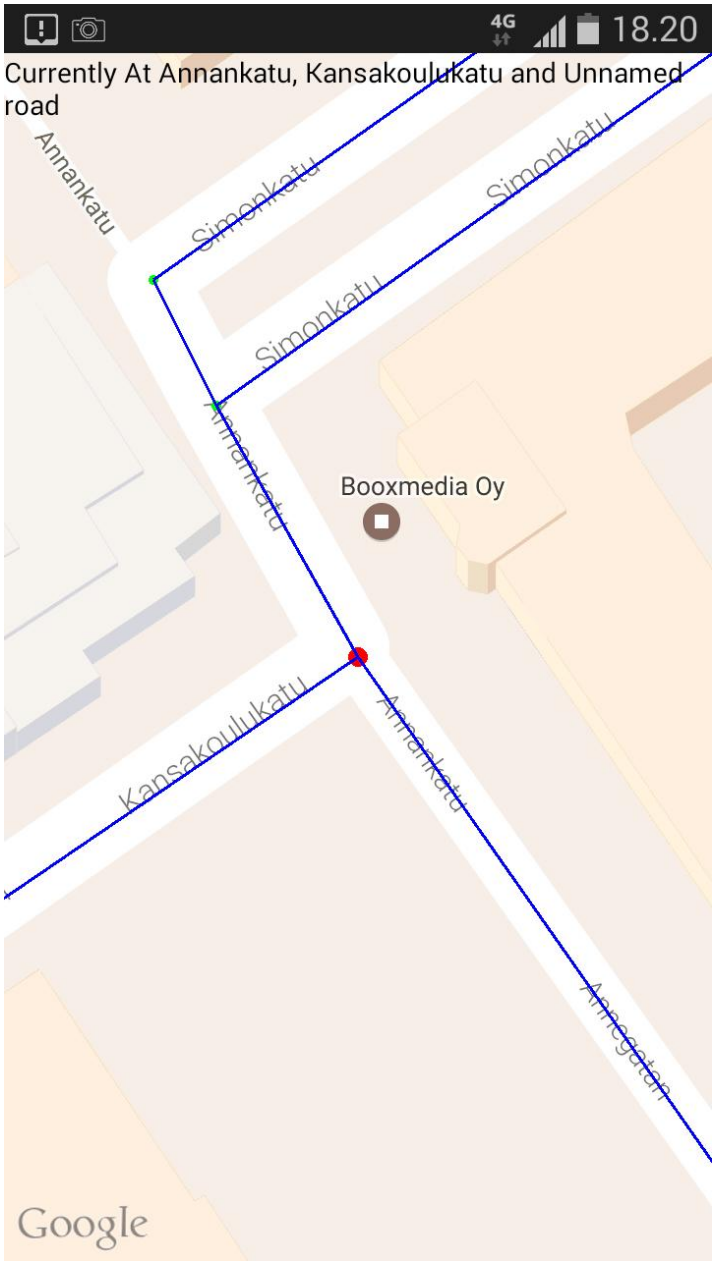

Figure 4. Intersection Explorer

Guide Dots (http://guidedots.org/) is a navigation application for visually impaired. It speaks the current address. The user can add and find places of different category. It has some similarities to WalkyTalky. The user has to register for the full use of the application. Figure 5 illustrates the interface of the Guide Dots application. When the application starts, the location is given verbally. In this example it says "You are currently at Annankatu 30".

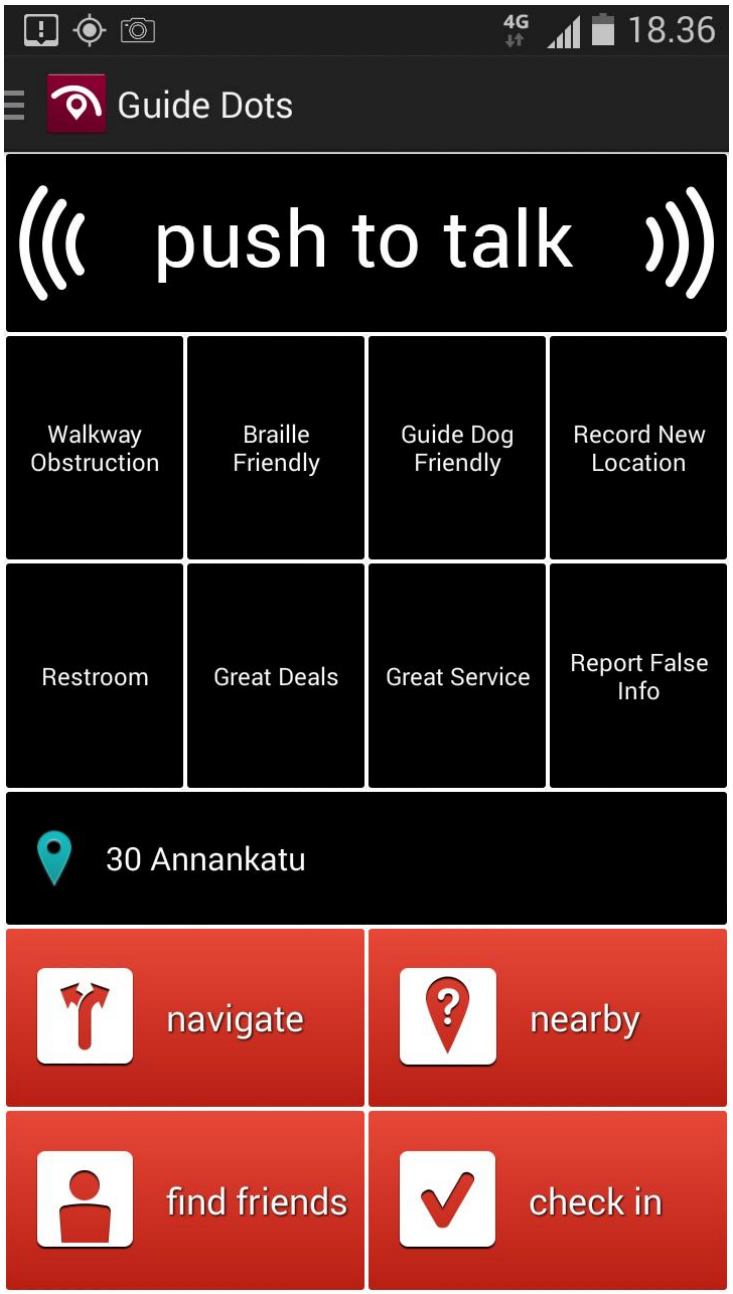

Figure 5. Guide Dots

The user can also use applications meant for regular users. This is possible because there is an application called TalkBack (https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback&hl=fi). TalkBack is an application developed by Google. It assists a visually impaired person to navigate other applications. For example, TalkBack reads out loud which button was pressed.

2.5 Computer Vision

Computer vision has developed intensely during last decades. It has been developed using software engineering and mathematics. Physics, psychology and neuro sciences have also contributed there.

Mathematically a camera is a plane in three dimensional space. Points in three dimensional space are projections on this plane. Points on a plane can be described as vectors. Vectors can be described as matrices. One of the most important matrices is the Fundamental matrix. Fundamental matrix can be created by combining points of two planes. The points of these planes mean the same point in 3d space. Figure 6 below illustrates the situation.

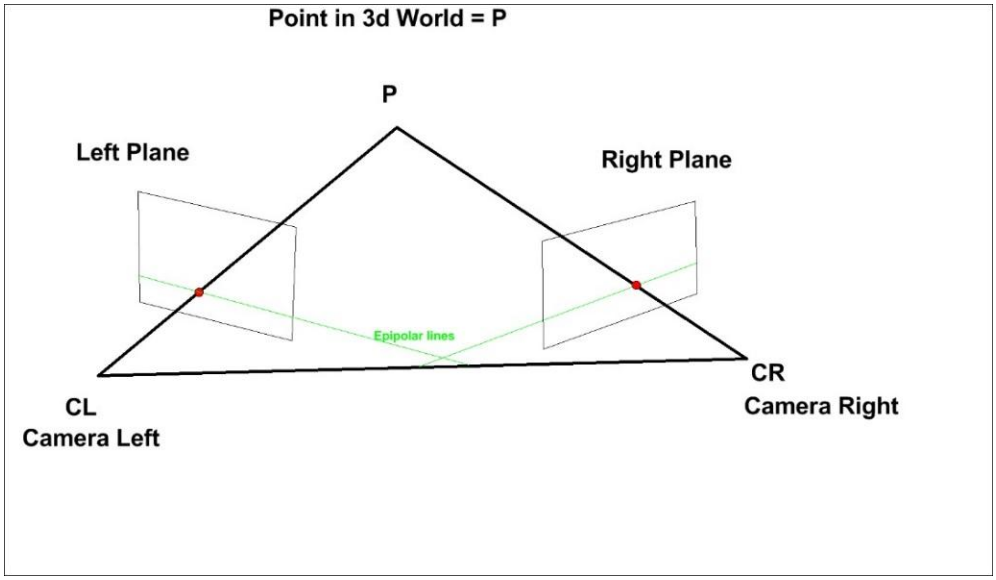

Figure 6. Points on planes

In Figure 6 there is the epipolar plane. It is the triangle which is drown between the two cameras and the point in the 3d world. There is an epipolar line on each of the planes. Epipolar line is drown between the cameras. Using the epipolar triangle it is possible to calculate the corresponding dots on each plane for each point in the 3d world. These dots are red in the image. The dots are usually described as vectors. They are the x and y values of the dots on the planes.

The Fundamental Matrix is used to describe the points as vectors. Fundamental matrix can be described using the following equation:

$$x'^{n}Fx = 0$$ [23]

In this equation x' and x are the same point in 3d projected on two planes. In the equation ^{n} means transposed.

The points on the planes can be considered as vectors from the original point in 3d. The points can also be used as tensors.

The other important matrix is called Essential matrix. Essential matrix can be calculated directly using Mat_<double> E = K.t() * F * K of the OpenCV. Essential matrix is Fundamental matrix, but in Essential matrix 0<x<1. Fundamental matrix provides a way to calculate the movement of the camera between the two images. Fundamental matrix binds the movement of the camera with the following equation:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^{T} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}.$$ [23]

Measuring distances using the camera of the phone appears to be straight forward operation. This is not so for some practical problems.

The first problem is to find the same points in the two images or planes. There are several methods to do this. SURF is one of the most used. SURF comes from Speeded-Up Robust Features. SURF uses vectors, which in an application are data structures, like matrices. If the vectors of an area in an image differ only little, the area is considered the same in both images.[24]

Another practical problem are the points falling outside the previous image. The camera has to be rather stable. If the camera rotates or moves too much, the equations fail.

The third problem is that the lines of the epipolar plane usually do not meet at the point P. There are several reasons for this. The lens of the camera can be the cause. Also the points on the planes are actually defined using masks. If the masks are big, which means there are a number of pixels under the mask, the lines cannot be pointed accurately. These problems were also considered by Hartley and Zisserman (Chapters 9-12, Multible View Geometry in Computer Vision, http://www.robots.ox.ac.uk/~vgg/hzbook/). There they explain the concept of minimization of geometric error. They suggest multiple methods of triangulating the 3d point of the Epipolar plane.

SURF is not used in the applications of the present project. The method used is called Optical Flow. Errors happen when finding the same points in the two images. The functions of the OpenCV library ensure the points in various ways between the images. Then for example the colors of the pixels are compared.[25]

Optical Flow uses information of the previous images. It tries to guess the location of a point by calculating the movement of the point on a plane. The equation below describes the process. The aim is to calculate the difference of the coordinates in two images in relation to time:

$$I(x,y,t)=I(x+dx,\ y+dy,\ t+dt)\ .$$ [25]

The above equation can be simplified using Taylor series:

$$I(x,y,t) = I(x,y,t) + (\partial f / \partial x)dx + (\partial f / \partial y)\mathrm{d}y + (\partial f / \partial t)\mathrm{d}t.$$ [25]

By removing $I(x,y,t)$, the result is:

$f_x u + f_y v + f_t = 0$. [25]

The Optical Flow equations, like $v = -(f_x / f_y) * u - (f_t / f_y)$, can now be derived. This equation proofs that the solution is a line in the image or on the plane.

Below are the Optical Flow equations:

$$f_x = \partial f / \partial x$$
$$f_y = \partial f / \partial y$$
$$u = dx / dt$$
$$v = dy / dt$$. [25]

These show that if they could be solved, the results would be the speed of the point on a plane or the difference of the point on a plane. They also show that there are two unknowns, it cannot be solved as such.

There are some methods to bypass this problem. One method is Lucas - Kanade. In this method nine pixels are chosen (One and the eight surrounding it). This leads to 9 equations and 2 unknowns. In practice only two equations are solved per image per point. [25]

Because $A * d = b$ --> $(A^n A) * d = A^n * b$, where A^n = transpose A, there are two numbers for each point on the plane or image.

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$
$$A^T A \qquad\qquad A^T b$$

[25]

Optical flow equations are discreet. The derivatives usually mean values from pixel to neighboring pixel. The difference is between those pixels in x direction and y direction. Gradient vectors are calculated in similar fashion. Gradients are used in edge detection. Edge detection is used in obstacle avoidance application, which is described in Chapter 6.

2.6 Location, Object Recognition and Collision Avoidance

Finding the current location, object recognition and collision avoidance were considered the key parts of the project. As mentioned earlier, there are many applications for locating. The other two key points are under rapid development.

Triangulation is the basis for distance measuring. To locate, triangulation is necessary. To avoid collisions distances have to be calculated. Triangulation has been used for locating for a long time. It is based on two observations of the same object. The object, and the observations form a triangle as illustrated in Figure 7. The observations are made at alfa and beta corners. If the distance between alfa and beta corners is known and if the angles alfa and beta are known, all other sides of the triangle can be calculated. This means that the distance to the object from alfa corner or beta corner is known.

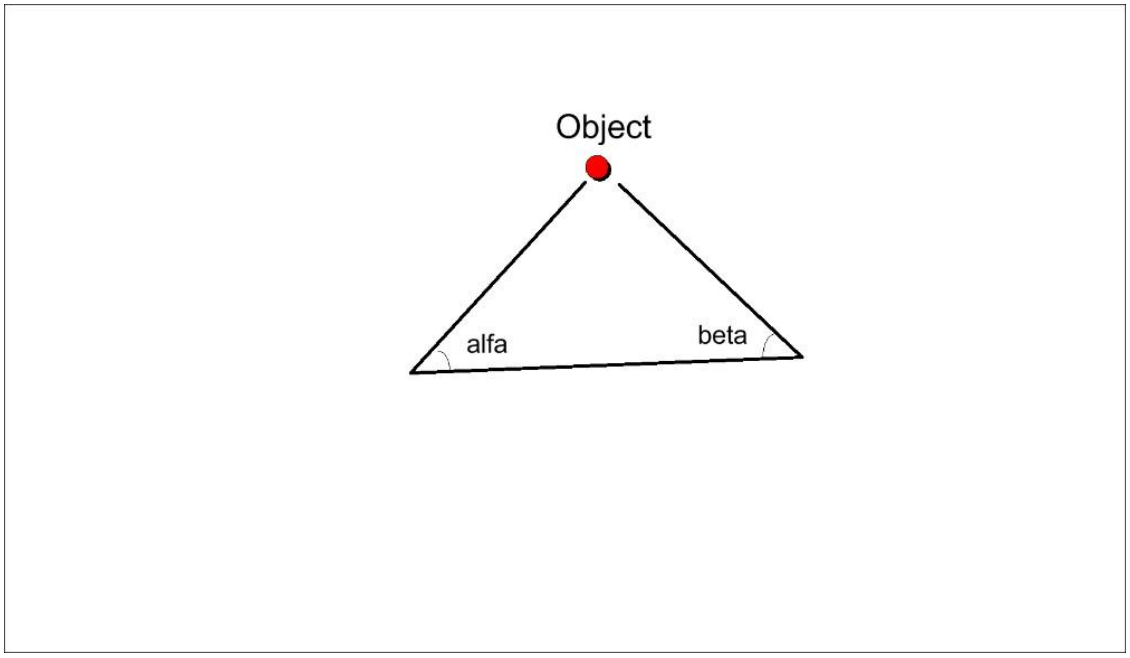

Figure 7. Triangulation

Triangulation was made visually in the past. It was used to map terrain. Nowadays radio signals or laser are used. GPS is based on differences in times of the satellites, but triangulation is still used there.

Object recognition is easy for a human in general. For a computer it is a challenge. Several methods have been developed to overcome this challenge. Appearance based methods use for example color and shape of the object. These can use edge detection

and grayscale matching to determine color and shape of the object. Feature-based methods are also numerous. These can use data structures called trees to store large amount of information about the object. Usually the tree stores information about the object in different angles and lightings. Genetic algorithms are the third common method for object recognition in computer science. These are in the realm of artificial intelligence. Some good results have been achieved. These are not described in this paper, because they are not in use in any of the applications used in this project.

Collision avoidance is based on sensors. A computer collects data from sensors and reacts to inputs. Autonomous cars are an example of collision avoidance. These cars usually use radars and laser beams to measure distances to obstacles.

Visually impaired could benefit from this rapid development of object recognition. They would like, for instance, the phone to be able to tell the contents of a fridge. Because the present project aimed at helping moving around, these whishes were rejected. Object recognition was applied in application SeeZebra (SeeZebra is not yet available at Google Play). This application announces "Zebra" when it detects the traffic sign for pedestrian crossing. This application is described in detail in 4.3.

There are applications for object recognition. They usually are not meant for visually impaired. There is an application which recognizes coins and bills. This has been developed for visually impaired. Hopefully applications like this will be more numerous and variable in the future. An example of these applications is EyeNote (http://www.eyenote.gov/). These applications are useful even if coins and bills can be recognized by touch.

Collision avoidance using the phone is the main subject here. The interface did not turn out to be an issue, because there is ready to use text to voice interface. This means that the phone can guide the user verbally. Collision avoidance is meant to be applied while the user moves. Mathematics of present dictates that 3d structures can be constructed using one camera, if the camera moves. Minimum of 7 points in each image are needed. [28] Mathematics also dictates that if the application uses lines, at least three images are needed. The mathematics are described later in this paper.

The current application in the set is SeeObstacle1 (SeeObstacle1 is not available at Google Play). SeeObstacle1 uses laser distance measurer to aid detecting obstacles.

The main reason for this approach was the battery life of the distance measurer. They last for over a year in daily use.

3 Examples for Solutions

After handling the background of the project it is time to talk about the actual solutions for using smartphones in the context of aiding the visually impaired to move around. There are some applications which help to avoid collisions. These are discussed first. There are many other approaches to solve these problems. One approach is to use additional hardware. Examples of these are Structure Sensor and GuideCane. These are also described and briefly discussed. Then some aspects of the actual software project are discussed. Some approaches to the problem were omitted. These are explained. The outcome of this process is discussed in chapter 4.

3.1 Collision Avoidance by Smartphone

University of Alicante has developed a 3D phone based application for obstacle detection. [29] Using the cameras the developed software is able to measure distances to obstacles that the user encounters. A 3D phone often has two cameras which allow obtaining the view in stereo. The application uses stereo pairs of images to calculate the distances to obstacles. The application extracts 30 000 measures per frame. It handles nine frames per second. The user interface beeps and vibrates to alert the user about the obstacles it finds.

In figure 8 there is an example of a 3d phone. It has a pair of cameras which make it possible to record 3d videos. There are 3d phones which have only one camera. These phones can display 3d images. It is possible, that the application could not be run on the phone model in Figure 8. Manufacturing the models used for developing the application has ceased. Whether the application would run or not depends on for example the Android version of this smartphone.

Figure 8. A Dual Camera Phone

The application also uses other sensors of the phone. It uses magnetometers and accelerometers to determine the orientation of the phone and the direction in which the user is walking. The application uses large buttons which are easy to use for visually impaired.

An advantage of this solution is that all the technology is embedded in a smartphone. There are various auxiliary devices that could be used to enhance the solution. There are additional sensors that detect obstacles. The sensors use a variety of technologies to achieve this. To use these sensors require some maintenance. The user has to for example recharge the batteries of the sensor device. Carrying auxiliary devices can also be cumbersome.

The application starts to detect obstacles from the start. This means that no previous calibration is required. This is one more feature of the application targeted for visually impaired. Using cellular phone is now common. This helps the visually impaired to comfortably blend among people. The application warns the obstacles via acoustic signals or vibrations without depriving the user from the sense of hearing. The application is meant to be used to complement the use of a cane.

Manufacturing of the phone that the software was developed for has ceased. The phone had binocular vision enabling the application to estimate the distance to an obstacle. The obstacles are detected at the distance of 2 meters from the phone. The audio gets louder as the obstacle approaches. The application cannot pinpoint the direction of the obstacle. It only warns that there is something to avoid nearby. [8]

In 2013 Daniel Koester conducted a study on guidance and obstacle evasion software for visually impaired. In his study he mentions that real time information handling as one of the challenges. The paper describes a dual camera system which is based on gradient calculation. The software finds an accessible path for the user based on similarity between pixels in the area of the camera view.

He also mentions a GuideCane. GuideCane is a cane with wheels at the other end of the cane. Using sensors the cane guides the user around obstacles. In Figure 9 there is the GuideCane. It uses light sensitive sensors. It tries to avoid dark, because it is then probably under an obstacle. It uses wheels to drag the cane to where the sensors indicate no obstacles.

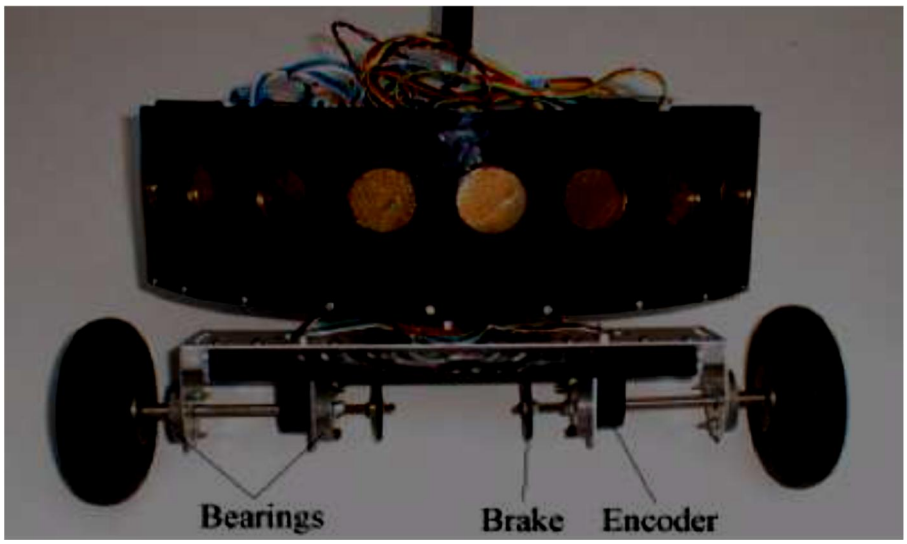

Figure 9. The Other End of GuideCane

Daniel Koester proposes a modular software framework as an answer. Modular framework allows many developers to contribute simultaneously. Modularity allows the best solution to be applied for each module of the framework.

A modular software can be applied in many ways. This means that the modules can be used in some other appliances. For example obstacle avoidance could be used in a traffic appliance. Gathering information about environment can be used for several purposes.

There already are several frameworks. These include robot operating system, navigation toolkit and augmented reality framework. Using modules means that there must be communication between modules. This can slow down the system. Multithreading helps to speed up the system.

The developed framework for visually impaired uses mostly C++ language. Multithreading is supported by C++.The framework is encapsulated into a library. The library offers several interfaces for the programmers. Modules in the framework can be configured. New modules can be added to the framework. Modules can be grouped into pools.

In the paper the framework has been used to create ground detection system. The orientations of surfaces can be obtained using the system. The system tries to find accessible regions on the ground. The system works well if stereo vision can be achieved. [9]

There are some applications available, which use the camera to find obstacles. Usually they use sound to warn. The applications which I was able to find, had some issues. The meaning of the sounds was sometimes difficult to interpret. It takes some time to learn what each sound might mean. If the user moves in a familiar environment, interpreting the sounds is easier. One of the programs I tested was Voice for Android. [2]

The vOICEforAndroid(https://play.google.com/store/apps/details?id=vOICe.vOICe&hl=fi) application scans the screen and gives audio feedback about possible obstacles. It takes some time to learn what the different sounds mean. The voices are based on

brightness and the height of the bright object. The brighter the object is, the louder the sound. A small object sounds like a beep. The Voice for Android does not support depth mapping for obstacle detection on Android phone.

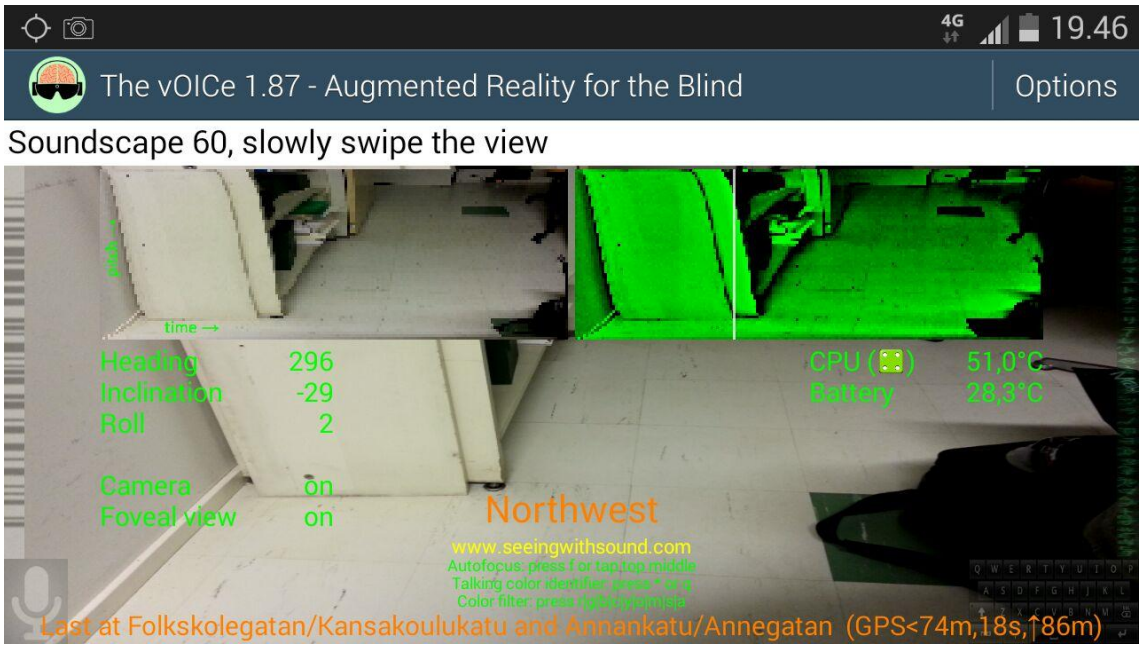

Figure 10. Voice for Android

The user can use the haptic feedback of the application by swiping slowly across the screen of the phone. The phone vibrates when the user's finger is on a bright object on the camera screen. The application can be used with camera classes with earphones. The classes are connected via a USB cable. Turning the head is easier than turning the phone camera for finding bright areas.

The Voice for Android has many features. The application can identify colors. This feature turns the torch light of the phone on. For the color identification to work, good light conditions are needed. The talking compass speaks the current heading. The talking locator speaks nearby street names. Talking face detector speaks the number of faces detected in the live camera view. Shake the Voice allows the user to initiate user-specific action. Shake the Voice is part of the Voice for Android application. Speech recognition is also supported by the application. Tapping the right edge of the screen launches Google Goggles application. Google Goggles recognizes printed text using optical

character recognition. Tapping the left and right edge of the screen can be configured to launch the user's favorite applications.

Testing this application proved that interpreting the sounds is difficult. A productive way to use this application was the recognition of colors. In a familiar environment this helps to navigate, especially indoors. After the user learns the colors of the environment, hearing which color is in front is useful. Using color codes to mark routes or objects could be useful.

These outside applications can be used with the applications of the project. For example the user can turn the SeeCompass on and find which way he or she is facing. The user can also use the laser distance measurer to find obstacles. If the user is in an area where there is traffic, it is useful to find pedestrian crossings. Using all applications is more convenient turning the TalkBack on. Some users prefer not to use TalkBack. (https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback&hl=fi).

Safety of the user was the most important criterion for the applications. This aspect influenced mostly the development of the camera application, but it also affected the development of the other applications. Initially the camera of the phone was meant to recognize the obstacles. This proved to be too challenging a task for the time provided for the project. Safety of the user also affected the decision to reject the location based application from the project. The location coordinates (GPS) deviated from the actual coordinates which caused the guidance to fail. [31]

3.2 Structure Sensor

Testing the Structure Sensor differed from the three previous applications. The reason for this is that the source code was available. The project began with the idea that obstacles could be detected by the system. The first experiments were done using Structure Sensor (https://store.structure.io/store). Structure Sensor is a laser scanner. It can be connected to a computer or a phone using USB cable. Using the right versions of NDK, OpenNI and Python it works on Linux phones. The tests were made using a portable computer. Distance measurement worked well using the Sensor. For example,

Structure Sensor ships with an application which measures the distance to the nearest object. This application could directly be used to warn about objects.

The reasons to reject the Sensor were the small details in using the Sensor in practice. The Sensor can be attached to an iPhone as shown in Figure 11. Figure 11 shows how the Structure Sensor connects to an iPhone. Structure Sensor was tested on a Windows portable computer.

Figure 11. Structure Sensor on an IPhone

This possibility was not available for Android phone. However, there now is the possibility to 3d print a bracket to hold the Structure Sensor. This option was not there when this decision was made. The user had to attach the Sensor somehow and that proved to be problematic. Manufacturing an own rack was considered. That would only work on a specific Android phone model, which led to rejection of the idea.[32]

There were also other issues with the Structure Sensor. The Sensor uses rechargeable battery, which has to be charged almost every day. For any user that repetitious task

could seem laborious. If the user wants to use the Sensor, it is obligatory to recharge the battery. A visually impaired person has to find the recharger and the Sensor to recharge the battery. There are ways to make it easier to find objects, but these may not always work. The battery could also run out while the user was using the system. This could lead to a situation where the safety of the user was in jeopardy. All of these factors led to the decision to use some other way to detect obstacles. [1]

3.3 Development Process

One aim of this project was to make use of an agile process. The users were closely involved in the development process. Several meetings were held at Iiris, the headquarters of Näkövammaisten keskusliitto. In those meetings the requirements evolved. Some applications were discarded and some were inserted. Some applications were upgraded.

During the project different kinds of methods were used to keep track of the development. The user stories were stored using pen to a notebook. These included "to do" lists. Usually one application was discussed at a time. Several applications were developed almost simultaneously, because the applications were modular. The development of the applications was not delayed due to insufficient requirements. There were other projects, which delayed the development.

The feasibility of the project is valid. The overall objective of the Näkövammaisten Keskusliitto is to help visually impaired cope better with their condition. The project aims to do the same. Visually impaired use phones regularly, so, the applications are easily integrated in the used system. The three applications can be used simultaneously with other applications.

Functional requirements evolved during the process. This is true for all three applications. This is also true for the applications which were not in the chosen set. Both groups of applications are discussed in this chapter.

The next step in obstacle detection after the Structure Sensor was to try using the camera of the phone. Using the camera would drain the battery of the phone faster. This means that the phone has to be charged more often. The users, however, are used to do that. They use phones regularly for other purposes. The fewer gadgets the user has to find, the easier it is for them to use them. The applications presented in Chapter 3.1

had not been tested for all environments. No applications which use multiple cameras were tested. For a visually impaired person it could be difficult to tell if it was dark or not. The camera does not work the dark. This kind of observations led to studying situations where the camera would give misleading results for the user.

There are many other situations where the camera gives unexpected results. There may be something, like a finger of the user, on the lens of the camera. This prohibits the camera from functioning correctly. It is difficult to find an algorithm, which could warn the user of this. Many other things could cover the lens. Sometimes the camera is out of focus even when the lens is clear. In some phone models out of focus of the camera can be detected by the hardware and software of the phone. This allows the programmer to include warnings in the application of this situation for the user. The phone used for this project had hardware for camera focus. The software could not access the value for the focal length, which is possible in some other phone models. Using this model it is possible to set the focus permanently for maximal distance in an application. This is important when measuring distances using the camera. If it rains, the phone may be damaged permanently. The rain may also obscure the vision of the camera. It is also difficult for a camera to distinguish between an image and real world. This may cause confusion in some applications. Windows cause reflections and a camera application will probably interpret the situation wrong. Each of these situations should have a correct response by the application. It was observed that this cannot be achieved using just a phone and an application. The user should be warned about this prior to using the application.

Some other ideas to use the camera came up during the development process. One of these was to guide the user to stay on a pavement to prevent getting hit by a car. The application did not perform correctly. The application interpreted as the side of pavement all things which have sharp edges. This led to misinterpretations by the application. Also, the application could guide the user to a driveway because it too has sides. There were attempts to improve the performance of this application, but the security of the user could not be guaranteed.

It is possible to use the camera to measure distances. This requires that some aspects of the images are known. For example, knowing the height of the camera from the ground makes the measuring more accurate. Using the other sensors and that the camera moves are also helpful. For this project measuring distances was not possible to do in a

way that would be safe for the user. This remains one of main future development challenges for the project at hand.

During the process it frequently happened, that because the camera was not safe to use in many situations, using extra equipment could help to make using the smartphone safer. Using extra equipment has pros and cons. In this case using the laser meter to measure distances proved to be more accurate than using the camera. The reliability of the system was the decisive factor in favor of the measurer. The measurer requires using batteries, which was considered initially to be cumbersome. The batteries only have to be changed less than once a year. The measurer is more robust than the phone. It has been designed for professional construction environment. It works in a dark environment. The user can hold the phone safely in a pocket.

While testing it was observed that in an urban environment there are many windows and reflective surfaces. This makes measuring with the laser distance measurer more challenging. Fortunately the users usually move in a familiar environment, where they know where this kind of errors may occur.

A different way to interact with the user had to be considered. The emphasis had to be on verbal interaction, but using the touch screen was not totally rejected. Response times of the user and the phone were taken in account. In the following image the process of interaction is described. In the image the user inputs the system and the system responds verbally. The user may use ear plugs. The TalkBack [4] and S Voice [3] applications help the user to start the applications.

Figure 12. illustrates the process of Interaction. The user has started the application and moves. The application analyzes the collected data, the signals, and speaks to guide.

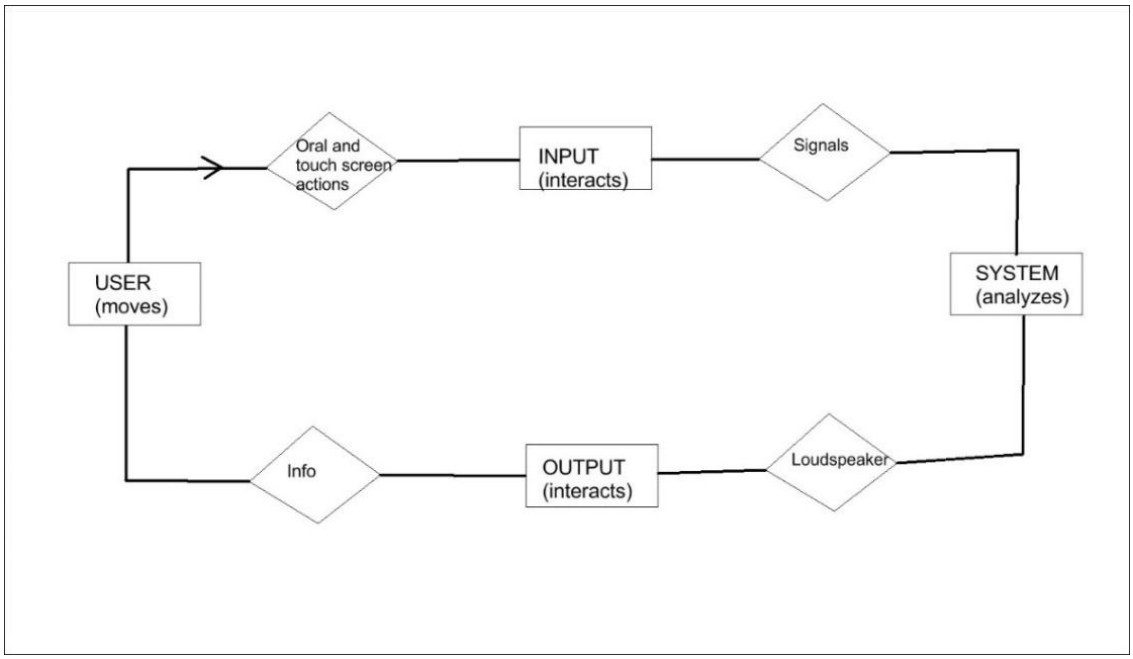

Figure 12. User Interaction

The aim of the project was to allow a visually impaired person to move around in an urban environment. It was agreed that knowing the direction the user is facing could be useful. Using the laser distance measurer in a certain way was considered a way to detect obstacles. Finally, it was considered useful for an application to recognize a certain traffic sign, zebra crossing. These three applications are the outcome of the project.

The camera application was initially meant for obstacle avoidance. This requires that the camera can detect distances. This led to the need to use NDK. NDK uses the resources of the phone more efficiently than Java and Dalvik. It soon turned out that an outside libraries were needed. The libraries found were OpenCV[6] and PointCloud[7]. Especially OpenCV proved useful.

Using all the found tools were not sufficient for this camera application. One of the major reasons for this was that the camera moves irregularly. It is difficult to know how much the camera has moved since the camera last pointed at same direction. This leads to unsolvable equations. If it was possible to keep the camera pointing at the same direction, the problem could perhaps be solved. Some efforts were made to try this, but it turned out that the stability of the phone was the key factor for the application to work. The frame rate was too slow for the application to keep track of the points of interest. In other words the points of interest could not be followed from frame to frame. This problem

can be solved using the other sensors of the phone. The angle that the camera is pointing can be resolved. This gives the key to solving the problem. If the angles provided by the sensors change too much, the frames provided by the camera can be omitted.

The location based application started as navigation application. There are some good applications for this purpose for the visually impaired. This led to rethinking of the application. One suggestion was that when the user arrives at a location the user could record a voice announcement which the phone would read out loud when the user arrives at the same location again. If the location based application works well this announcement application is not needed. When testing these location based applications it was observed that the accuracy of the locator was not good enough for example for negotiating a junctions. Sometimes it worked well, but sometimes the accuracy deviated from previous trials. These facts led to the conclusion that the available applications for map based navigation offer satisfactory service for their requirements. To pinpoint the location exactly is a problem for all of them. To pursue developing a pinpoint accurate navigation application, which this project required to be safe, was considered not necessary.

SeeCompass application was considered safe to use. There are situations when the application may not work, but those situations are adequately covered. In those situations the phone fails to announce the current orientation in degrees. There are two buttons on the touch screen: One is for hearing the degrees, the other is for Help. Pressing the Help button causes the phone to read out loud the main points to use the application. The Magnetometer of the phone seemed to work well under almost all circumstances. If the user points the phone straight up, the system may fail. The user soon learns how to hold the phone for the application to work. There is another version of this program. This version can be stopped using verbal command: "Stop". Initially this was considered a good option for an interface. The benefits for the user to use voice commands proved to be minimal. The touch buttons are easy enough to find.

Laser distance application is not as safe as the SeeCompass. The laser beam may miss the obstacle. Glass surfaces may reflect the light beam, which causes the measurement to fail. In these situations the user does not know that the measurement has failed. The user has to learn to use this aid. The user usually learns where this system works and where it fails. The laser distance measurer application announces the distances. The accuracy of the announcement is one tenth of a meter, when the measurement does not

fail. This application was considered the best of the available choices for the purpose of detecting obstacles.

To buy the laser distance measurer and install the application correctly must be done by an outsider. The application uses Bluetooth to detect the distance measurer. It makes a list of found Bluetooth devices. The application should always find the right Bluetooth device. This process needs an outsider. Once this has been done correctly, the application always chooses the right device if the distance measurer is turned on.

The application which detects zebra crossings has not changed a lot during the development process. When the traffic sign is in the view of the camera, the application announces "Zebra". Pointing the camera of the phone at different directions the user learns where the zebra crossing is. This application has not been tested extensively. For example it has not been tested under darkness. During the testing the camera gave false announcements about traffic crossings, but mainly these happened indoors.

In some instances the development process took too long. Trying to measure distances using only the camera of the phone proved too difficult. Developing this application took too long. The decision to reject this application should have been made sooner. Some time was also lost on location based application. In this case also the decisions should have been made sooner. There were also some other deviations from pure agile process. Daily meetings did not occur. Working in pairs was not an option in this case. Scrums and sprints did not occur frequently and regularly enough.

The applications were tested using Samsung GT –N7105 smartphone. The phone has 1.6 GHz processor. The size of ROM memory is 2.0 GB. The display / touch screen is 7 cm wide and 12 cm high. The display has 1280 * 720 pixels. The physical size of the phone is 151.10 * 80.50 * 9.40 mm . The phone weighs 183.00 grams. The phone was released 25. of October in 2012. It has 3100 mAh battery. The phone uses Android version 4.2.2. It can be updated to Android 4.3 Jelly Bean.

The phone has the standard sensors. The camera has 8 Megapixels. The front camera has 1,9 Megapixels. Location sensors of the phone can be used for various applications.

The connectivity of the phone is good. It can use 4G network. It can also use Wi-Fi. Bluetooth is also available. It uses GPS. Users can also transfer data using Micro USB.

All three applications should work on a wide variety of Android phones. This was not, however, tested.

It was initially agreed that at this state the main function of each application would be sufficient. This means that the development of additional features like options and a help system will be developed later.

The process produced three safe and working applications and some applications which could be developed further and added to the set. Most important of these is the Structure from Motion application using the camera of the phone. The current applications are described in the following chapters.

4 Three Applications

This chapter is presents the three working applications developed for this project. Orientation of the user is tackled first. This is why application SeeCompass is in 4.1. After this in 4.2 collision avoidance is discussed. SeeDistance1 is the application for this. The third application in this chapter is SeeZebra, which collects information about the environment.

4.1 SeeCompass

The accuracy of the phone's GPS does not allow any more exact localization. [33] The accuracy was not sufficient for locating pedestrian crossings, which was one of the goals for this project. The goal was to pinpoint the exact location of the pedestrian crossing in relation to the user. If the error the phones locator makes is 2 meters, the phone may guide the user totally wrong. This became clear while testing the developed application. The application was assessed to be hazardous to the user. Some other approach for location based application was needed. The location aid for visually impaired in this project was decided to be SeeCompass (https://apkpure.com/seecompass/com.hos.seecompass). First, the user interface of the application is introduced. Figure 13 shows how the interface looks like after the user has started the program. It is a snapshot of the touch screen. Pressing Degrees button causes the application to say " One hundred and five" in this example.

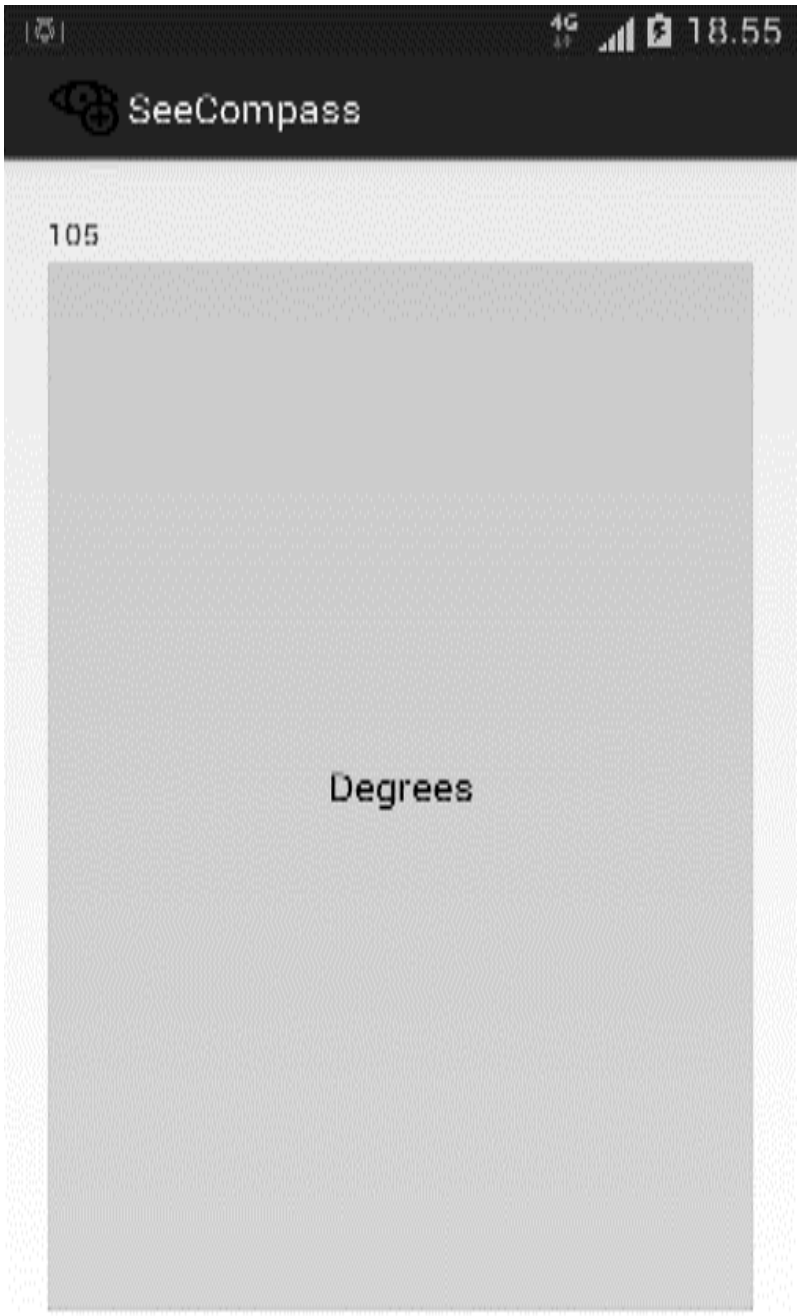

Figure 13. The User Interface of SeeCompass

The user can start the application using S Voice[3]. S Voice is an application by Samsung. It is a personal assistant based on Vlingo. S Voice listens to the user and answers verbally. S Voice has 18 key words. For example, the user can ask for weather forecast by saying "What is the weather like today". The application recognizes the word weather and reacts accordingly. Other words to use are Call, Message, Task, Music, Text, Look up, Navigate, Record voice, Hands-free mode, Schedule, Search, Open app, Timer,

Memo, Read the news, Turn WiFi on, and ask a question like "What is the fastest car in the world ?".

S Voice can be activated using a keyword. This key word can be chosen by the user. Once S Voice has been activated the user starts The SeeCompass by saying "Open SeeCompass", where Open is the key word and SeeCompass is the name of the application to start. The user can also start the S Voice by pressing the S Voice icon on the touch screen.

Once the SeeCompass has started it starts taking the bearings using the methods for sensor events listener -interface. Using this interface it is possible to read the sensors of the phone when the values of the sensors change. The phone has a built in magnetic sensor for this kind of applications.

TextToSpeech class of Android API was used to make the application announce the bearings of the phone to the user. TextToSpeech has 7 methods. Using these methods a programmer can control for example the rate and pitch of the speech. A programmer can also stop the speech with these methods. Not all of these were used in the application. The bearings are stored in a text field when the user presses the Distance buttonr. Then addSpeech method of the TextToSpeech class reads the text out loud.

The magnetometer of the phone must sometimes be calibrated. This is not a feature of this application, it is a feature of almost all Android phones. There are plenty of instructions on the internet of how to set the bearings right. The phone has to be moved in a certain way. There are different moves for different phone models. The fact that 0 degrees or 360 degrees does not always point exactly to the north, is not as catastrophic as one would first think. If the phone is used in a familiar environment, the degrees are also familiar. It does not matter as long as they stay the same. This is how this specific phone operates: Its 0 degrees is almost in the east, but it stays that way. If it is calibrated it soon returns to its former state. If the SeeCompass application is used, it is not to be calibrated. This guarantees persistent performance and it is easy to find bearings.

Voice commands were considered to be used to stop the application, but using them proved to be difficult in a noisy or windy environment. As long as the microphone detects voices that are loud enough, the application will not terminate. Sometimes this could take minutes.

This version of the application does not work on all models. The upgrade for new smartphones will be done later. The Android version must include TextToSpeech class. This class requires the phone to reside within the reach of a mobile network operator.

4.2 SeeDistance1

This application has two activities. When the application launches it searches for nearby Bluetooth devices. If the application finds the distance measurer among these devices it chooses it and connects to it. The programmer has earlier read the id of the distance measurer and the application uses this id to connect successfully. When the connection is operational the second activity starts.

The second activity listens to the Bluetooth messages from the laser distance measurer. When a user presses the Bluetooth button of the distance measurer device, the measured distance is sent to the phone. The phone receives the sent value and turns it into desired units. The values are temporarily stored in a text field. TextToSeech class of Android API is then used to read the value in the text field. addSpeech method then reads aloud the value of the text field using the audio devices of the phone.

This application uses the same text to speech interface as the SeeCompass application. The announcements are read out by a female voice. The user can adjust the volume of the voice by using the volume adjustment set of the phone. It was considered not necessary to use voice commands with this application either.

No interface is provided here, because the user interacts with this application by opening it and listening to the distances.

4.3 Distance Measurer Device

Leica Disto D3a BT [5] is used to measure distances. This device has a lot of features. The way to use this device with the Distance Measurer application is to push the start button until the device beeps. This means that the device is on continuous measurement mode. In this mode distances are measured continuously. When the user pushes the

Bluetooth button next to Start button of the laser distance device, the current distance is sent to the phone.

The user has to start the measurer first and thereafter the application. Running the application is easy. The user only has to press the Bluetooth button to hear the measured distance. Pressuring the button multiple times and aiming the device to different directions the user is able to map the surroundings. The measuring can be stopped by pressing the Start button again. Figure 14 shows the distance measurer. The Bluetooth button is the blue button next to the red measuring button.

Figure 14. The Distance Measurer

The Distance Measurer has many functions. Only the basic measurement and Bluetooth are used. In Figure 14 there are some measurements on the display. The red button is the one used to switch measuring modes. The same button turns the measurer on.

4.4 SeeZebra

The SeeZebra application is based on Haar features cascading. [34] Haar features cascading means that that a computer has been trained to recognize a certain object. Training has been done using hundreds of photographs of the object. In this case the object is the zebra crossing traffic sign. Training process produces a cascade xml file. The file is a tree structure. Feature recognition is done with the help from this file.

Haar features are edges, lines or center-surround. There are four edge types that are recognized as features. Eight types of lines are also recognized. Center-surrounded types there are two. These three features are recognized in a frame.

Cascading means that the same region of the frame is inspected several times using layers of the tree in the cascade file. If all layers contain similarities with the original photo then pixels in the area are points of interest. Each layer has different set of elements to recognize. These elements are lines, edges and center-surround.

The CameraView activity is an interface CvCameraViewListener2. This activity links the OpenCV library to the camera of the phone. The CameraPreviev.xml file contains the camera view object. The object is provided by the Android API. CameraView class draws a blue rectangle around the detected traffic sign.

The CameraView class uses the Detector class to detect zebra crossing traffic signs. The Detector class uses the Java.io objects to read the cascade.xml file. Once the cascade file has been read, the Detector class uses the detectMultiScale method of the cascadeClassifier class to detect the features of the photo in the frame retrieved from the camera. This method returns the detected objects (features) as a list of rectangles.

SeeZebra application sometimes makes mistakes. It interprets some other objects as the pedestrian crossing traffic sign. This will have to be dealt with in the future. For now, the user has to use additional information, like orientation and location to safely use SeeZebra.

Figure 15 shows SeeZebra in action. The application has found the right traffic sign on a computer screen. It repeatedly announces "Zebra" and shows the found traffic sign.

This image was created by using a n image capturing application for Android phones. This is what the user of the SeeZebra application sees.

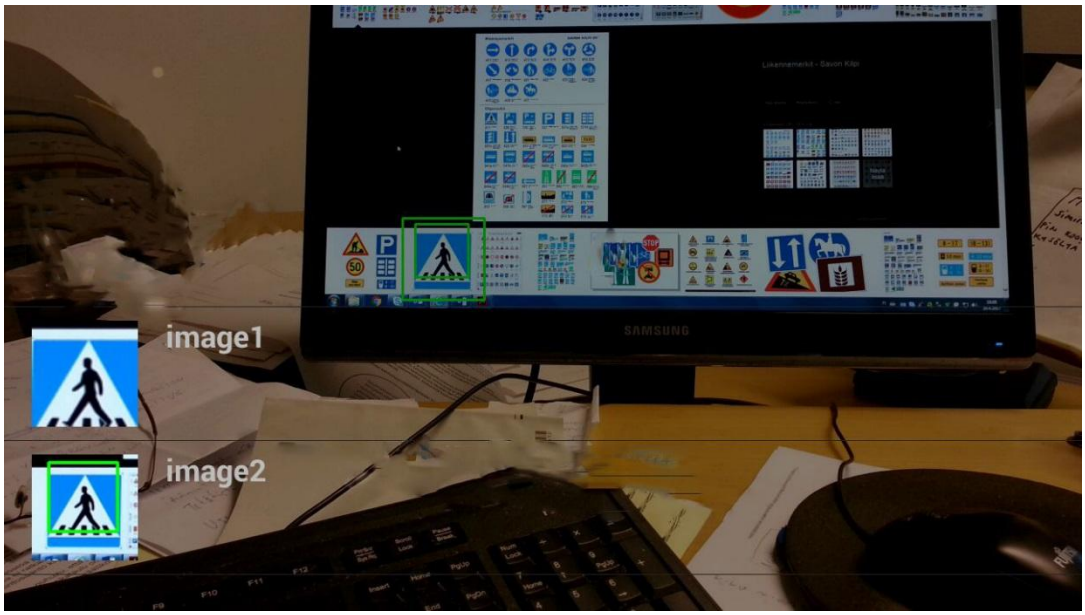

Figure 15. The SeeZebra Application

The SeeZebra application has in the figure above found the right traffic sign among many possible other traffic signs. Application found the sign even if it was displayed on a computer screen.

5 Additional Libraries, OpenCV

Additional libraries provide a huge advantage when developing applications. They are well tested. The most used library during this project was the OpenCV. It is used in the camera applications. OpenCV is used in SeeZebra application. This application calls the functions of OpenCV from Java.

There is a fourth application not mentioned earlier, which belongs to this set. This application was developed to replace the laser measurer application SeeDistance1. This new application uses the camera to measure distances, therefore it uses the OpenCV. This is described in Chapter 6.

OpenCV algorithms work with the camera. It has also algorithms for example for sound, because using the camera includes recording videos. These algorithms cover nearly all

one can do with a camera of a phone. Here it is discussed how OpenCV can be used with Eclipse SDK to develop applications for Android.

OpenCV is a modular library. The Core module consists of data structures. The most important for the applications is the Mat. Mat is a multi-dimensional array. It is usually used to store the pixels of an image. Analyzing images would be difficult without Mat. Core also includes the basic functions used by other modules. Imgprog is a module for image processing. These processes are for example image filtering and image transformations. Video is a module for video motion estimation, background subtraction, and object tracking algorithms. There are modules for both single camera and several cameras usage. There is a separate module for object detection. Highgui is an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities. These are some of the modules of the library.

OpenCV enables automatic memory management. This means for example allocating and de allocating memory for matrices. Multiple functions can use the same matrix. The library makes memory usage efficient in many other ways too. For example the library uses fixed array types. These are array types that are best suited for image and video processing. These arrays can be used from several threads simultaneously.

Using the library requires installing OpenCV Manager on the phone. OpenCV Manager contains most of the OpenCV code. OpenCV does not run on Dalvik (Java) Virtual Machine as Java applications do. It runs on native code. The application uses the library interface with the help from helper function. This helper function binds the application with the OpenCV Manager services. All this happens hidden from a programmer.

A programmer has to remember to add camera usage permissions to project's Manifest file. This is necessary even when the camera is used using native function calls. OpenCV allows to use java camera view.

CVCameraVieListener2 is an interface for camera view. This interface has functions like Start(), Stop() and Frame(). For example The Frame() function defines what happens when the camera produces a frame. The programmer has to define in that function what happens at that moment.

In the layout file the programmer has to add the java camera view widget. In the activity code the programmer can find the widget by find by id method. Now the OpenCV camera listener should work. The frame is wrapped to an OpenCv Mat -type function. Mat is an OpenCV multidimensional array. This function returns frames of different types. The programmer has to define the type of frame that is returned.

OpenCV has C and C++ interfaces. A programmer can choose which to use. C offers the usual data types. It also offers the possibility to create other useful data structures like binary trees. Binary tree is useful in search algorithms. C++ uses classes. Class structures are used both in Java and C++ languages. Using class structures has many advances. This is the reason C++ is preferred instead of C.

All applications have the same necessary classes to make the camera applications to work. The OpenCV library must be installed in the used workspace. After starting a new android project, the library must be added to the project. The necessary classes to use the camera with OpenCV are The Activity, BaseloaderCallback, JavaCameraView and OpenCVLoader.

Activity is the basic class for all Android applications. It has a life cycle. An application may contain many Activities. An activity has a set of methods to determine the different actions during each step of the life cycle.

BaseLoaderCallback is an OpenCV class. It takes care of the service binding of the OpenCV methods. It listens whether an OpenCV service is needed by the activity. It also listens the different stages of the service. The programmer actually creates a child class of the BaseLoaderCallback by using the java key word new.

JavaCameraView is a handler object for Java camera view. JavaCameraView is an OpenCV class. It handles the input from the camera of the phone, so, it acts as an interface of the camera.

Finally, the OpenCVLoader object binds the OpenCV with the Activity object. OpenCVLoader must be activated at the OnResume() stage of the activity's life cycle.

The programmer has to add necessary permissions to Manifest.xml. These include permissions to use the camera. These permissions are mostly the same as for using the camera through Java interface.

In this project OpenCV was included in many applications. These applications were aimed to collect information about the environment. This information was about detecting different aspects of environment.

One application was aimed at finding doors. This application was based on edge detection of OpenCV. This application was developed to a level, which was not adequate for this project.

There was also an application developed to detect the sides of a pavement. This application was similar to the previous one. It used the same OpenCV algorithms. This application just found the edges were at the lower part of the image.

Another application in this category was the one to measure distances to obstacles. This application used the optical flow algorithms. These algorithms track specific pixel areas from frame to frame. The specialty of these algorithms is that they try to predict where the pixel area will be in the next frame. These algorithms are very efficient.
Yet another camera application suggested was an application to recognize bills. This application would have used the same algorithms as the traffic signs recognizer. This application was never tested.

6 Future Work

There are a number of things to do in the future of this project. These include adding new applications to the set, factoring the current applications and releasing the work to the public. In this chapter discuss these matters are discussed.

This project will continue to factor all three applications. The SeeCompass will be the first to change. The SeeCompass application will have more options which are chosen using voice commands. In other words the user will be able to verbally make selections. This requires a more extensive help system. This system will explain what each spoken

command will do. The user should be able to choose the accuracy of the measured direction. Sometimes it is sufficient to announce the four main bearings "north", "east", "south" and "west".

The appearance of the user interface will change. There will be a common logo for all applications of the set. Most likely there will be an eye in the logo represented in some form.

Factoring and developing the distance measurer application will also include a new Help system. This application could also have several options for the accuracy of the measurements. These also could be chosen verbally. This application could also be implemented using a camera. The idea would be that the camera should remain relatively stable while walking. A stable camera would allow the application to use algorithms like optical flow of the OpenCV library. A stable camera would make it possible to find the points of interests from frame to frame. The optical flow part of the application is ready. Applying it so, that it can securely measure distances, will have to be developed in the future. The name of this application will be SeeDistance2. In this scenario the laser distance measurer would be obsolete. For now, the laser distance measurer will be used.

Development of the traffic sign application will be carried on. It will recognize the zebra lines of the crossroads. Possibility to recognize other aspects of traffic environment will be researched. For instance, the poles and the structures that facilitate traffic lights are of standard design. The application could recognize traffic lights this way. Using OpenCV functions it is easy to make applications recognize poles. In the future it may be wise to make the application first recognize the poles and only after that start looking for the pedestrian crossing traffic sign. This would eliminate errors, which occur from time to time. These errors mean that that the application has identified something else as the searched traffic sign. Programming the application in this way would improve the safety of the user.

The first application to replace one of the three applications will be the distances measuring camera. There are number of reasons to replace laser distance measurer. The laser distance measuring application is easy to use once the connection building

with the Bluetooth is automated. The automatization cannot be done by a blind person. The automatization requires the id of the laser measurer. This is due to the Bluetooth protocol. This means that the application cannot be downloaded directly from the Google Play. The other major reason is that there should not be additional gadgets besides the phone.

Some parts of this application are already working. This application measures distances to obstacles at the sides of the user. It warns with a sound if the obstacle is too close. The obstacle must have corners. Smooth and curved obstacles are ignored. The idea is that if the user walks too close to a wall the user will be warned. This application is based on the fact that the camera is usually at the height of 1.5 meters. The application draws lines where it finds edges. The application searches the lines only from a certain areas of the frames. The lines must be found at the lower half of the frames. Also, the lines must be oriented towards where the user is going. If these conditions are met, the distances are measured.

The measured distances cannot be trusted. If the obstacle has not sharp edges on the ground, it will be ignored by the application. If it is dark, the application should warn the user, because the application does not work properly. In Figure 16 it is shown how the application found edges and colored them red.

Figure 16, Measuring distances

To measure distances using 3d construction the single camera has to move. This movement cannot be wobbly. The height and the aim of the camera should be maintained during the motion. This contributes to application being able to calculate the motion of the camera between two frames. OpenCV API provides means to extract geometric structures from images taken from a moving camera. This is sometimes called the monocular approach. This approach tries to achieve the same result as two cameras achieve in depth vision. The goal is to find where the objects of the scene are in relation to the camera. This is done by the process called triangulation. Triangulation is usually achieved by ray intersection. Ray intersection means that there are two rays, one from each camera, that intersect where the object is located. This is not, however, a reliable way to triangulate an object. There are several ways to triangulate.

The OpenCV version the author used does not contain an API for triangulation. This has to be achieved using C++ functions. To achieve triangulation there must always be at least two views. The views can be taken from two cameras simultaneously or as in this case two images from a single camera from two different positions. Using OpenCV functions it is possible to calculate how much the camera has moved. This is the key to 3d vision using a single camera. There must be at least seven points which are recognized as the same in both images. Fortunately there is a function in OpenCV library to help calculating the motion of the camera. The function is FindFundamentalMat. This function is explained later.

First thing the application has to do is to find the same points in two consecutive images or frames. OpenCV provides several ways to do this. Optical flow is one of them. It has been developed during several years and as a result it is both fast and reliable to use. Optical flow is a method for matching selected points from one image to another. The images must be relatively close to each other. Using the brightness constancy it is possible to subtract on image from another. The subtraction results a zero if there has not been any changes between the images. The subtraction makes optical flow more efficient. There are number of other methods that make optical flow produce better results. Using optical flow with OpenCV is relatively easy using function calcOpticalFlowPyrLK.

For the purpose of this application it may not be the best solution. OpenCV's extensive framework for feature-matching has matured. Some of them are explained here. A descriptor is a vector of numbers which describes the surroundings around the point of interest. OpenCV has several methods to construct the descriptor. These methods use different data structures. Below Code Example 1 is an example to use these methods:

Code Example 1.

```
// 1. detectingkeypoints
SurfFeatureDetector detector();
vector<KeyPoint> keypoints1, keypoints2;
detector.detect(img1, keypoints1);
detector.detect(img2, keypoints2);

//  2. computing descriptors
SurfDescriptorExtractor extractor;
Mat descriptors1, descriptors2;
extractor.compute(img1, keypoints1, descriptors1);
extractor.compute(img2, keypoints2, descriptors2);

// 3.  matching descriptors
BruteForceMatcher<L2<float>> matcher;
vector<DMatch> matches;
matcher.match(descriptors1, descriptors2, matches);
```

Detecting the points of interest is the first step in Code Example 1. This is done using the OpenCV function SurfFeatureDetector(). This function finds the points of interest. Each point of interest is represented as a point on a plane. These points construct the vector KeyPoint.

The second step in Code Example 1 is to form the descriptors from the points of interest which were stored in keypoints1 and keypoints2. Data structure Mat can be used here to store the descriptors. As mentioned earlier Mat is an OpenCV matrix. The main reason to use this is that it speeds up the process.

The third step is to store the matches. They are stored in the vector DMatch. The feature matching process has been made conveniently by using descriptors. The BruteForceMatcher produces too many matches. Some filtering will have to be done in Code Example 2 there is an example of this.

Code Example 2.

```
// Check that the found neighbors are unique (throw away neighbors
// that are too close together, as they may be confusing)
std::set<int>found_in_right_points; // for duplicate prevention

for(inti=0;i<nearest_neighbors.size();i++) {
DMatch _m;
if(nearest_neighbors[i].size()==1) {
_m = nearest_neighbors[i][0]; // only one neighbor
} else if(nearest_neighbors[i].size()>1) {
// 2 neighbors – check how close they are
double ratio = nearest_neighbors[i][0].distance /
nearest_neighbors[i][1].distance;
if(ratio < 0.7) { // not too close
// take the closest (first) one
_m = nearest_neighbors[i][0];
} else { // too close – we cannot tell which is better
continue; // did not pass ratio test – throw away
}}
else {
continue; // no neighbors... :(
}
// prevent duplicates
if (found_in_right_points.find(_m.trainIdx) ==
found_in_right_points.end()) {
// The found neighbor was not yet used:
// We should match it with the original indexing
// ofthe left point
_m.queryIdx = right_points_to_find_back_index[_m.queryIdx];
matches->push_back(_m); // add this match
found_in_right_points.insert(_m.trainIdx);
}
} cout<<"pruned "<< matches->size() <<" /
"<<nearest_neighbors.size()
<<" matches"<<endl;
```

All these steps are implemented for the pairs of images. This is generally called triangulation in this context.

Edges are found using intensity of light. Different set of edges can be found using different parameters of the OpenCV functions. The parameters have to be tested for optimal performance of the application. This testing is being done. On the next figure is the same scene as in Figures 16 and 18. This figure shows the gray scale image after edge detection. Opencv operates using gray scale images, but the output may vary.

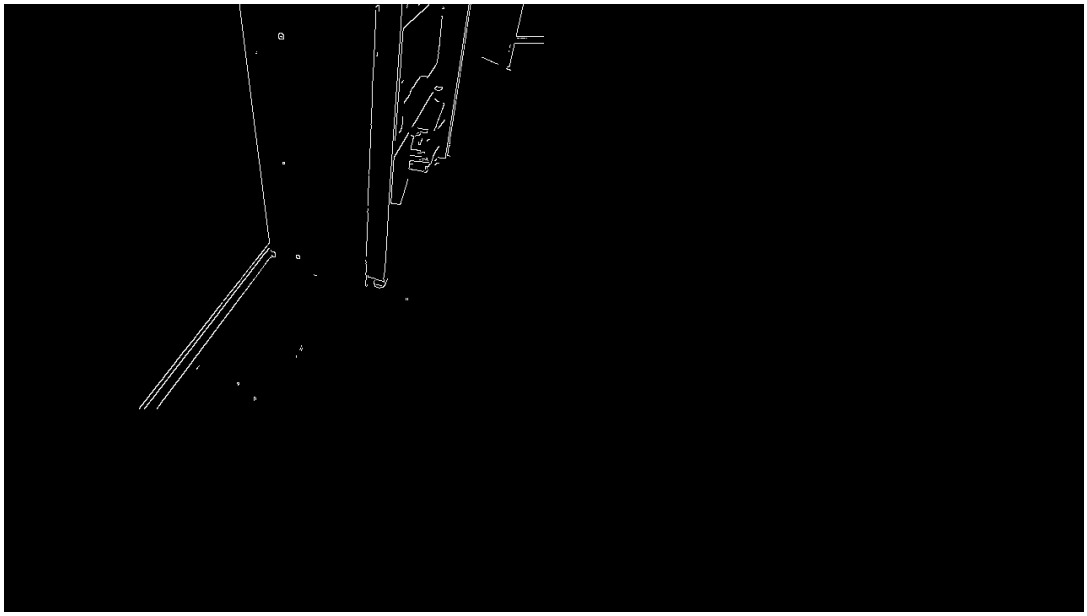

Figure 17, Edges

OpenCV usually converts color images to grayscale in this kind of applications. This ensures efficient execution of the functions. Also, usually integers are used, when it is possible. Usually edge detection requires filtering. Most used filter is the Gaussian filter. Filtering reduces noise. The idea here is that noise in the image is distributed randomly. This application uses Canny edge detection.

The application has C++ functions to detect edges. There are many options in OpenCV library to detect features. There are also many options to detect edges. These include Sobel, Scharr, Laplacian filters and Canny edge detector. For example Laplacian edge filter produces lines that are seldom straight. Canny edges produce clean lines from the detected edges. Canny edges is affected more by the random noise in the camera frames which can causes the lines to change a lot between images. This depends on the environment the application is being used.

The image is transferred to grayscale by the function cvtColor. This function turns the frames from the camera into CV_RGB2GRAY format. Then the program sets the region of interest to lower half of the image. This is where the sides of walls usually are in the image. Then the program uses Hough Transform. This is what the OpenCV documentation says about Hough Transform "Hough Transform is a popular technique to detect any shape, if that shape can be represented in a mathematical form. It can detect the shape even if it is broken or distorted a little bit". OpenCV has method HoughLines to detect lines. The program uses this method.

The program has C++ class LineFinder. This class uses other C++ classes to calculate the distance of the phone to the side of the pedestrian path. This class also draws the lines to indicate the sides of the road. It also eliminates those lines which are not probably sides of the road. Some of the calculations are performed in this class. LineFinder has method stringFromJni which a callback function from Java side. This function returns the calculated values in string format to Java.

To help distance measuring optical flow is being used. In the following figure the application uses Optical Flow of the Opencv.

Figure 18, Using Optical Flow.

In Figure 18 the scene is almost the same as in Figures 16 and 17. The phone has moved so that the vertical angle of the camera has increased. The camera has not moved forward or backward. The red lines indicate the movement of the points of interest. The length of each line can be measured. When the camera moves back or forward (without changing the angle) those lines, which are closest are the longest, because they have moved the most measured in pixels. Knowing these facts can already solve part of the distance measuring problem. In this situation the user has to move the camera first backward or forward. During this the lengths of the lines have to be calculated.

To calculate distances the image must be devided in sections. The areas close to the sides of the image can be used when going forward. If the camera stays relatively stable, long lines close to side of image means that an obstacle is close. However, this method will fail, if the obstacle is infront of the user. Other methods must be applied in that situation.

The solution to that situation is that the edges, which form horizontal lines are recognized. The application already recognizes edges that form vertical lines. The application already calculates distances to those lines and warns about them.

Solving this problem is not difficult, because the current application prohibits finding any other lines than almost vertical. If this restriction is removed all edges are found by the application. If the camera points forward and the user moves forward, the distance to the line formed by the edges can be calculated. The other sensors of the camera are not yet used by the application. If it turns out that accelometer and gyroscope must be used, they will.

Location based applications will most likely be left out for now. Setting the language correctly the current available applications can be made speak Finnish. One more reason to leave location based application is the usage of batteries. There may be differences between models of phone, but the consumption of batteries increases substantially when using an accurate locator.

7 Conclusions

The customer or the user is at the center of focus when programs are developed using an agile development system. Customers should affect every step of the development process. This was one aim of this project.

Choosing the number of users involved at each step of the process was not simple. This was an important aspect especially at the beginning of the project since it determined the overall direction of the project. Ideally, there would be as many users involved as possible. In practice this number had to be allocated to fit the resources. This problem was partially solved by using representatives of the users. When the applications are in general use in Google Play it is possible to receive feedback from wider audience.

Meetings with the customers were not frequent enough. The meetings affected the development of the applications significantly despite of this. Some of the desires of the customers had to be rejected. Demos and collecting user feedback about the newest version of the program worked well. It has not yet been possible to meet with the customers at Iiris about the very latest versions. The most significant comments concerned the user interface and the options there had to be for each application. The changes to be made in the future are based on the customer interviews made earlier.

Some advances were made during this process. The SeeCompass application is available at Google Play and at some other applications sharing sites. Perhaps the idea of naming the application in this way will become clearer to the users after the other applications will appear there. The idea is to collect seeing aids at the same place. Using compass point to guide can help a visually impaired person in some situations. This was one of the main ideas of the project.

The SeeZebra works well enough for another round of testing. To recognize objects using the phone is not easy. If the user has some idea of the location of a pedestrian crossing the application can now guide the user to the exact spot. One of the four basic ideas was to use the camera of the phone to gather information about the environment and use that information. This application is an example of this. The camera was also used

to gather information for the application SeeDistance2. Obstacle avoidance using a single camera is almost succeeding. The code is there, but needs more testing. Both applications will be developed further.

Obstacle avoidance can be achieved using SeeDistance1. Measuring distances is accurate and fast using the option of continuous measuring of the Disto laser measurer. This application is useful in an environment where there are steep drops. Indoors it can guide along corridors.

The fourth idea, the user interface was solved using three separate applications. The end user just starts the application and there is only one button to press to use the SeeCompass. The other two applications the user just starts and listens to them.

Refactoring is still needed. Modularity of the programs can still be increased. Using the single camera of the phone for obstacle avoidance is not ready. There are many publications which address this problem, but these are not usually meant for Android and Eclipse.

Using a single camera of the smartphone for obstacle avoidance will be easier in the future. The OpenCV library has Structure from Motion API for version 3.2.0. This will mean less code to write for the developer. The phones are also developing. This means that the algorithms will execute faster.

References

1 Occipital. Structure Sensor. http://structure.io/developers. Accessed 24 March 2017.

2 Google. VOICe for Android. https://www.seeingwithsound.com/android.htm. Accessed 24 March 2017.

3 Samsung. S Voice. http://www.guidingtech.com/27428/s-voice-guide/#c. Accessed 25 March 2017.

4 http://www.androidcentral.com/what-google-talk-back. Accessed 25 March 2017.

5 https://www.youtube.com/watch?v=L2nhTB9iWKw. Accessed 25 March 2017.

6 http://opencv.org/. Accessed 25 March 2017.

7 http://docs.pointclouds.org/trunk/. Accessed 25 March 2017.

8 http://sgitt-otri.ua.es/es/empresa/documentos/ot-1308-deteccion-obstaculos-invidentes-eng.pdf. Accessed 25 March 2017.

9 https://cvhci.anthropomatik.kit.edu/~dkoester/publications/koester2013diploma.pdf. Accessed 25 March 2017.

10 http://eyes-free.blogspot.fi/2010/10/walking-about-with-talking-android.html. Accessed 25 March 2017.

11 http://www.nkl.fi/fi/etusivu/nakeminen/julkaisu/nvrek_vuosikirja/1_4_arviot_nv_lukumaarasta. Accessed 25 March 2017.

12 13.3.2017, http://myweb.rz.uni-augsburg.de/~eckern/adp/history/historic-papers/1901_309_553-563.pdf. Accessed 25 March 2017.

13 13.3.2017, http://www.physics.uwo.ca/~mgc/Mat%20Sci%20notes-1.pdf. Accessed 25 March 2017.

14 13.3.2017, http://www.livephysics.com/physical-constants/optics-pc/wavelength-colors/. Accessed 25 March 2017.

15 13.3.2017, Isaac Newton By Michael White Page 170. Accessed 25 March 2017.

16 13.3.2017, http://srjcstaff.santarosa.edu/~lwillia2/42/WaveEquationDerivation.pdf. Accessed 25 March 2017.

17 13.3.2017, Einstein, A (1917). "Zur Quantentheorie der Strahlung". Physikalische Zeitschrift. **Pages** 121–128. *Bibcode*:*1917PhyZ...18..121E*. Accessed 25 March 2017.

18 13.3.2017, https://global.britannica.com/science/luminous-intensity. Accessed 25 March 2017.

19 13.3.2017, http://micro.magnet.fsu.edu/primer/digitalimaging/concepts/concepts.html. Accessed 25 March 2017.

20 http://www.nkl.fi/fi/etusivu/nakeminen/julkaisu/nvrek_vuosikirja/1_4_arviot_nv_lukumaarasta. Accessed 25 March 2017.

21 Uusitalo, M. A., Virsu, V., Salenius, S., Näsänen, R. & Hari, R. (1997) Activation of human V5 complex and rolandic regions in association with moving visual stimuli. *Neuroimage* 5:1053–59.

22 *Richard Hartley and Andrew Zisserman (2003). Multiple View Geometry in computer vision. Cambridge University Press. ISBN 0-521-54051-8.*

23 https://ase.tufts.edu/cogstud/dennett/papers/surprisesurprise.pdf. Accessed 25 March 2017.

24 http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html. Accessed 25 March 2017.

25 http://docs.opencv.org/trunk/d7/d8b/tutorial_py_lucas_kanade.html. Accessed 25 March 2017.

26 http://iopscience.iop.org/article/10.1088/1748-3182/9/2/025002/pdf. Accessed 19 April 2017.

27 https://www.photonics.com/Article.aspx?AID=42279. Accessed 19 April 2017.

28 https://www8.cs.umu.se/kurser/TDBD19/VT05/reconstruct-4.pdf. Accessed 19 April 2017.

29 https://www.youtube.com/watch?v=4pEw6YPFYAQ. Accessed 19 April 2017.

30 http://www.afb.org/afbpress/pubnew.asp?DocID=aw170208. Accessed 19 April 2017.

31 http://www.gps.gov/systems/gps/performance/accuracy/. Accessed 19 April 2017.

32 https://github.com/occipital/OpenNI2/blob/develop/README.md. Accessed 19 April 2017.

33 http://gpsinformation.net/main/altitude.htm. Accessed 19 April 2017.

34 http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html. Accessed 19 April 2017.