

master's thesis

# **Application of a Camera in a Mobile Phone for Visually Impaired People**

*Bc. Jan Hadáček*



May 2017

Advisor: Doc. Ing. Daniel Novák, PhD.

Czech Technical University in Prague  
Faculty of Electrical Engineering, Department of Cybernetics



## DIPLOMA THESIS ASSIGNMENT

<b>Student:</b>	Bc. Jan H a d á č e k
<b>Study programme:</b>	Open Informatics
<b>Specialisation:</b>	Computer Vision and Image Processing
<b>Title of Diploma Thesis:</b>	Application of a Camera in a Mobile Phone for Visually Impaired People

### Guidelines:

1. Consult existing papers on computer vision technologies applicable as assistive technologies for visually impaired people. Namely focus on robust OCR, object recognition and classification.
2. Design and implement a camera and image gallery interface for visually impaired people. Implement relevant assistive features (for example: voice labeling, horizon detection, face detection, zoom, high contrast...).
3. Verify the selected application on a sample of five visually impaired users.

### Bibliography/Sources:

- [1] Richard O. Duda, David G. Stork, and Peter E. Hart. Pattern classification and scene analysis. Part 1, Pattern classification. Wiley, 2 edition, November 2000.
- [2] Yangqing Jia et. al. Caffe: Convolutional Architecture for Fast Feature Embedding, Proceedings of the 22nd ACM international conference on Multimedia, 2014
- [3] L. Neumann and J. Matas, "Real-time scene text localization and recognition," Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, Providence, RI, 2012, pp. 3538-3545.
- [4] S. M. R. Bagwan and L. J. Sankpal, "VisualPal: A mobile app for object recognition for the visually impaired," Computer, Communication and Control (IC4), 2015 International Conference on, Indore, 2015, pp. 1-6.

**Diploma Thesis Supervisor:** Ing. Daniel Novák, Ph.D.

**Valid until:** the end of the winter semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic  
**Head of Department**

prof. Ing. Pavel Ripka, CSc.  
**Dean**

Prague, September 26, 2016



## **Acknowledgement**

Foremost, I would like to express my gratitude to my advisor doc. Ing. Daniel Novák PhD. for giving me the opportunity to work on this project. Besides that, I would like to give my thanks to Czech Blind United (SONS) and all the blind community, especially Michal Jelínek, Radka Komárková, Martin Procházka and Vladimír Krajíček for their valuable feedback.

A special thank belongs to my family for their tolerance and support.

## **Author statement for undergraduate thesis**

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in preparation of university theses.

In Prague, date 26.5.2017

.....  
signature

## Abstract

Tato práce se zabývá využitím kamery jako asistivní pomůcky ve speciálním telefonu pro nevidomé založeném na systému Android a vyvíjeném v rámci Katedry kybernetiky Českého vysokého učení technického v Praze.

V úvodu diskutuje uplatnění obecných úloh počítačového vidění v oblasti asistivních technologií pro nevidomé a zrakově postižené. Řeší jak specifické využití kamery (pro identifikaci objektů, bankovek, čtení textu), tak zpřístupnění základní funkce fotoaparátu a ovládání telefonu jako takového.

V rámci práce je implementovaný nový algoritmus pro rozpoznávání bankovek založený na BRISK deskriptoru a Gradient Boosted Trees klasifikátoru. Dále je rozebrána implementace přístupného mobilního uživatelského rozhraní k existujícímu rozpoznávači textu z reálných scén od společnosti Google a implementace jednoduché aplikace umožňující nevidomým a slabozrakým s pomocí telefonu pořizovat fotografie a pracovat s nimi.

Závěr práce popisuje uživatelské testování se šesti nevidomými a slabozrakými dobrovolníky, které v rámci vývoje výše zmíněných aplikací proběhlo. Aplikace vyvíjené v rámci této práce, se až na některé detaily setkaly u zrakově postižené komunity s pozitivním přijetím.

## Klíčová slova

mobilní technologie; asistivní technologie; počítačové vidění; ocr; rozpoznávání textu; rozpoznávání bankovek;

## **Abstract**

This thesis deals with the use of a mobile camera as an assistive tool in a special phone for the blind and visually impaired people, based on the Android system and developed in the Department of Cybernetics of the Czech Technical University in Prague.

In the introduction, it discusses the applicability of general tasks of computer vision in the area of assistive technologies for the blind and visually impaired. It deals with both the specific use of the camera (for identifying objects, banknotes, reading text labels), as well as accessing the basic functionality of the camera itself.

A novel banknote recognition algorithm based on the BRISK descriptor and the Gradient Boosted Trees Classifier is implemented as a part of this work. Implementation of an accessible mobile user interface to Google's existing real-time text recognition library, as well as the implementation of a simple camera and image gallery application is also discussed.

The conclusion of this thesis describes the user testing that took place during the development of the above-mentioned applications. The applications developed as a part of this thesis were (except for some minor details) received very positively by the blind community.

## **Keywords**

mobile technologies; assistive technologies; computer vision; ocr; text recognition; banknote recognition

# Contents

<b>1. Introduction</b>	<b>1</b>
<b>2. The “Core System” and other mobile accessibility solutions</b>	<b>3</b>
2.1. Characteristic of the <i>Core System</i>	3
2.2. The TalkBack service	5
2.3. Complete solutions	6
2.3.1. Kapsys SmartVision2	6
2.3.2. Claria Vox	7
2.3.3. Corvus	7
2.4. Specialized applications	7
2.4.1. TapTapSee	7
2.4.2. KNFB Reader	8
2.4.3. Darwin Wallet	8
<b>3. Specification of Computer Vision tasks relevant for accessibility</b>	<b>10</b>
3.1. Object recognition	10
3.1.1. Segmentation and Object Detection	10
3.1.2. Single class object recognition	11
Template based methods	12
Feature based methods	12
Methods based on convolutional neural networks	12
3.1.3. Multi-class object recognition	12
3.2. OCR / Text-in-the-wild recognition[29]	14
3.2.1. Algorithms based on image binarization	14
3.2.2. Algorithms based on edge detection	15
3.2.3. Sliding window approach	16
3.2.4. Algorithms based on deep neural networks	16
<b>4. Banknote recognizer implementation</b>	<b>17</b>
4.1. Functional requirements	17
4.2. Dataset	17
4.3. Pre-processing and data augmentation	17
4.4. Keypoint detection and description	18
4.5. Bag of Words	19
4.6. Classification	21
4.7. Geometric verification	21
4.8. Results	23
<b>5. OCR implementation</b>	<b>24</b>
5.1. OCR engines	24
5.1.1. Google TextRecognizer	24
5.1.2. Microsoft Cognitive - OCR	25
5.1.3. Comparison - summary	26
5.2. OCR user interface	27
5.2.1. Camera/document positioning mode	27
Confidence score	27
Upside-down detection	28
5.2.2. Text reading mode	28



5.3. Conclusion . . . . .	28
5.4. Examples of recognition results . . . . .	30
<b>6. Camera and gallery application</b>	<b>31</b>
6.1. Motivation . . . . .	31
6.2. Design of the camera application with assistive features . . . . .	31
6.3. Assistive features for image capture . . . . .	31
6.3.1. Face detection . . . . .	31
6.3.2. Tilt detection . . . . .	32
6.4. Labeling and saving of the image . . . . .	33
6.4.1. Blur detection . . . . .	33
6.4.2. Support for image recognition . . . . .	33
Microsoft Cognitive API . . . . .	34
CloudSight API . . . . .	34
6.4.3. Voice captioning . . . . .	35
Recording . . . . .	36
Encoding and storage . . . . .	36
Other metadata . . . . .	36
6.5. Design of an image gallery application with assistive features . . . . .	36
6.5.1. Browsing all images . . . . .	36
6.5.2. Browsing by category . . . . .	37
6.5.3. Browsing by date . . . . .	37
6.5.4. Low-vision filters . . . . .	37
<b>7. Testing</b>	<b>40</b>
7.1. Pre-test . . . . .	41
7.2. Tasks . . . . .	42
7.2.1. Text recognition . . . . .	42
7.2.2. Camera . . . . .	42
7.2.3. Pictures . . . . .	42
7.2.4. Banknotes . . . . .	42
7.3. Post-test . . . . .	43
7.4. Important findings of the testing . . . . .	43
7.4.1. Text recognition . . . . .	43
7.4.2. Camera and images . . . . .	43
7.4.3. Banknotes . . . . .	43
7.4.4. General findings . . . . .	44
7.5. Testing summary . . . . .	44
<b>8. Technical details</b>	<b>45</b>
8.1. Technologies behind the <i>Core System</i> . . . . .	45
8.2. Modules implemented as a part of this thesis . . . . .	45
8.2.1. The Banknote recognition engine . . . . .	45
8.3. Text Recognition . . . . .	46
8.4. Camera . . . . .	46
8.5. A list of third-party libraries used . . . . .	46
<b>9. Conclusion</b>	<b>47</b>

## Appendices

<b>A. Testing results</b>	<b>50</b>
A.1. Pre-test questionnaire . . . . .	50
A.2. Post-test questionnaire . . . . .	50
A.2.1. Subject M . . . . .	50
A.2.2. Subject P . . . . .	51
A.2.3. Subject H . . . . .	51
A.2.4. Subject V . . . . .	51
A.2.5. Subject S . . . . .	51
A.2.6. Subject R . . . . .	52
<b>B. Content of the enclosed CD</b>	<b>53</b>
<b>Bibliography</b>	<b>54</b>

# 1. Introduction

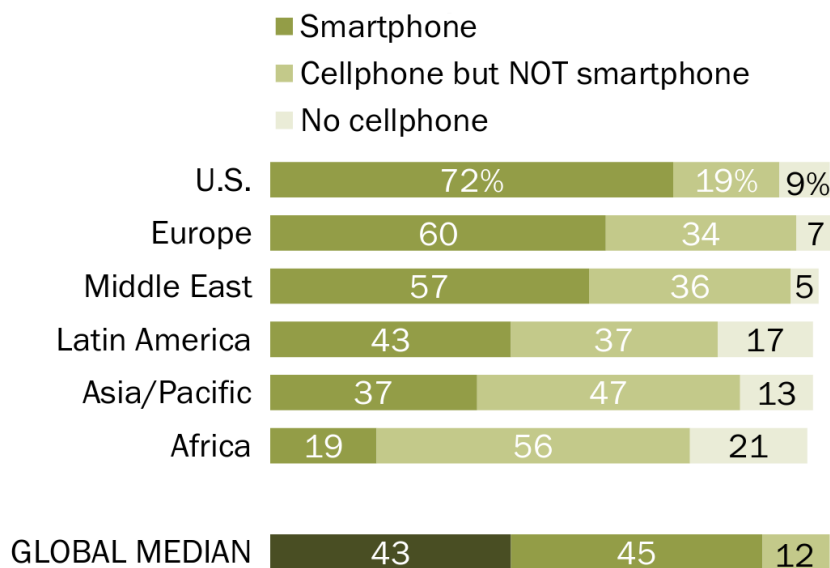
During the last decade, smart mobile technologies have become prevalent in everyday lives of ordinary people in developed countries as well as emerging nations. According to [1], the penetration of smartphones is growing in both developed and developing countries is estimated to be 68% and 43% respectively. The Figure 1 documents the spread of cellphones and the ratio of “smart” and “basic” phones in different areas of the world. So-called “smart” mobile devices are not only being used for communication. In general, many people tend to use smartphones to perform tasks that could only be performed on workstations and laptops in the past.

This work focuses on the usage of smartphones by blind and low vision people.

Blindness is the inability to see. The leading causes of chronic blindness include cataract, glaucoma, age-related macular degeneration, corneal opacities, diabetic retinopathy, trachoma, and eye conditions in children (e.g. caused by vitamin A deficiency). Age-related blindness is increasing throughout the world, as is blindness due to uncontrolled diabetes. On the other hand, blindness caused by infection is decreasing, as a result of public health action. Three-quarters of all blindness can be prevented or treated. [2]

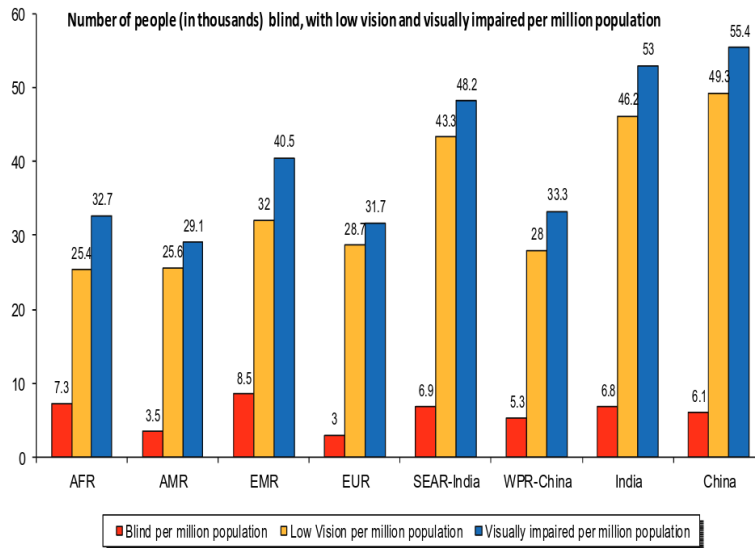
The World Health Organization (WHO) estimates that there are 285 million inhabitants with severe and uncorrectable vision issues, 39 million of which are legally blind [3].

While medicine cannot restore the sight of all people suffering from vision loss, assistive technologies can help them with their everyday tasks and improve their quality of life.



**Figure 1.** Regional medians of adults who report owning a smartphone/cellphone. [1]

## 1. Introduction



**Figure 2.** Blindness and low-vision prevalence in the world (thousands per million of inhabitants).[4]

By their nature, modern smartphones represent significant challenges to people with disabilities of this kind. As of 2017, smartphones with a hardware keyboard are almost nonexistent [5]. Touchscreen display and visually oriented user interface represent substantial obstacles, which is one of the reasons, why the blind and visually impaired community is more conservative than the general population in accepting these devices. For example in the Czech Republic, many blind people still use modified legacy Nokia phones as their primary communication devices.

The application developed as a part of diploma thesis by Petr Svobodník from Czech Technical University[6] represents an attempt to make popular Android-based smartphones accessible to people with serious vision loss. The solution presented in this work is designed to be a part of a system that was built upon the application (from now on called the *Core System*) and shares the same style of control with this system.

While the user interface of standard touchscreen smartphones presents a challenge to disabled people, increasing computational and storage capacity of mobile devices, as well as growing speeds and coverage of mobile Internet, provide unique possibilities for the use of cell phones as universal assistive devices. It is also important to realize that most people tend to carry their smartphones with them most of the times, so the accessibility tools integrated into those phones are more readily available than dedicated tools (i.e. color recognition devices, text reading devices).

This work focuses on assistive features that use the camera of the phone as their input, as well as on making the camera function itself accessible to users who are interested in using it to share images of the world around them with sighted people. It builds on existing work by Jan Pečan[7], who attempted to extend the *Core System* with some camera-based assistive features: namely banknote recognition and color indicator. The banknote recognition algorithm will also be revisited and improved as a part of this thesis in Chapter 4.

## 2. The “Core System” and other mobile accessibility solutions

This chapter discusses the design philosophy of the *Core System* and compares it to other mobile accessibility solutions based on Android. Like said in the introduction, the development of the the *Core System* started as a Master thesis of Petr Svobodník [6]. The basic design philosophy directly influences the design of user interface of accessibility tools suggested in this thesis. Some other contemporary systems mobile accessibility systems will be discussed for reference.

### 2.1. Characteristic of the *Core System*

The core philosophy of the *Core System* is to provide a user interface tailored to the target user group. Therefore, the user interface is very simplistic, minimalistic and even crude for conventional standards. No graphical elements are used at all, as the assumption is that the user utilizes mostly the sound feedback for orientation. Most of the time, the phone running the *Core System* only displays a short textual label in big font, which signifies current option in the menu.

The target hardware phone is a pure touchscreen phone; it only has three physical buttons (power + volume rocker). The use of these buttons is same as in other Android phones. All navigation in the menu is done using gestures performed on the touch screen. The whole structure of the phone user interface is a directed graph (almost a tree, with some exceptions). An excerpt of this menu structure is depicted in Figure 3.

The navigation in this menu is achieved using four touch screen gestures:

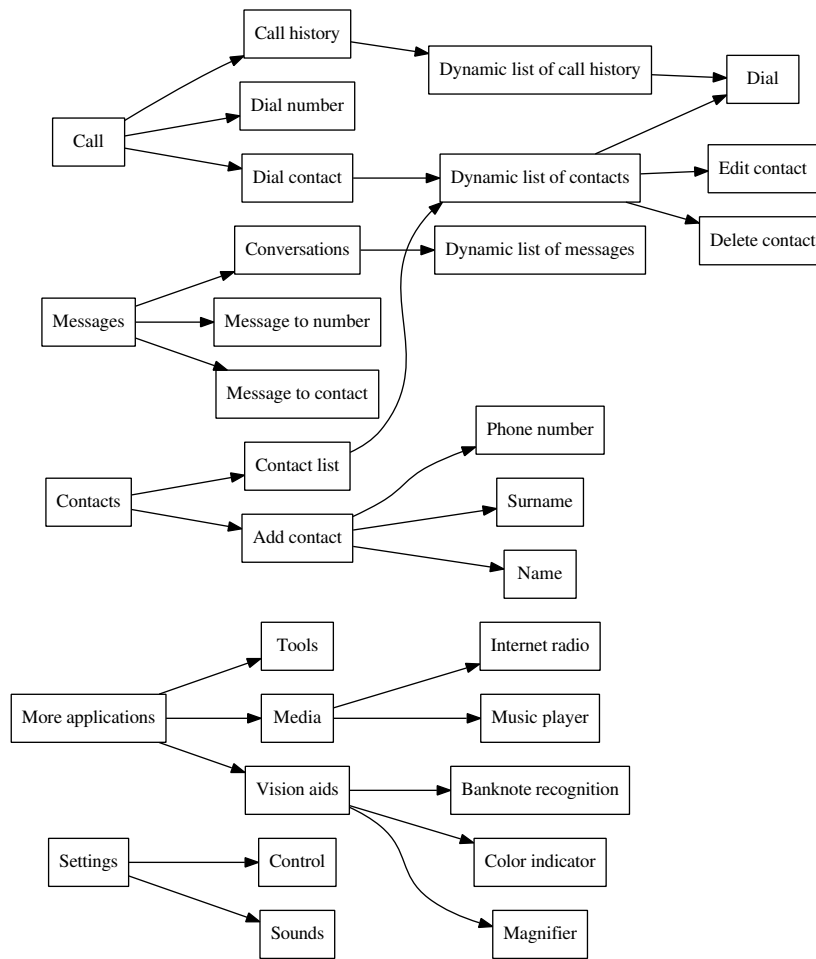
- tap in the left half of the screen (previous item in the list)
- tap in the right half of the screen (next item in the list)
- long press anywhere (next level of the menu tree)
- long press with two fingers (previous level of the menu tree)

During the navigation in the menu, the label of the current item and its position in the menu is announced (for example “Messages, two of nine”). It is very important to choose short and apt captions and, in the case of longer announcements, announce the most important points in the beginning, as more proficient users will choose to skip the announcement at any time (the voice output is interrupted by any action user makes).

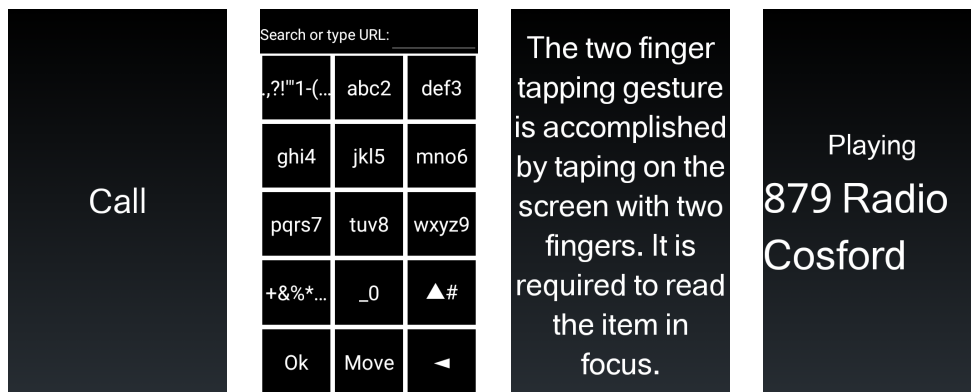
Text input is achieved using a touchscreen keyboard with voice feedback, as well as dictation input. An example of the user interface is shown in Figure 4.

The style of control should be consistent over the whole phone UI and the accessibility tools suggested in this thesis incorporate it. The *Core System* also supports third-party Android applications integrated via standard Android accessibility framework. Same gestures are used for native Android applications, which makes the control of them

## 2. The “Core System” and other mobile accessibility solutions



**Figure 3.** A short excerpt of first few levels the menu in the *Core System*



**Figure 4.** An example of the UI screens used in the *Core System*

somewhat more consistent with the *Core System*, albeit the very different nature of Android applications does not allow for perfect consistency in control.

The the *Core System* runs on dedicated low-end hardware with following parameters:

- SoC with a quad-core ARMv7 CPU - MT6580
- 512 MB of RAM + 256 MB SWAP
- 2+2 GB of flash storage (user + system)

- Android 5.1
- 3G (UMTS, HSPA, HSPA+) connection
- WiFi

All accessibility tools developed as a part of this thesis are supposed to run smoothly on this relatively constrained device.

The greatest advantage of the *Core System* is the specialized user interface. The main weakness is that only one device is supported and the device is rather low-end. Also, the support for third party applications is limited. Users are expected to use applications that are built in the *Core System* most of the time.

## 2.2. The TalkBack service

The TalkBack service is the standard way of making Android-powered touchscreen phones accessible. The strong point of this accessibility solution is that it is offered free of charge (often comes pre-installed on the phone already), it is open-source[8], and together with Android accessibility framework[9] (which is part of the standard Android system library[10]), it can make any compliant third party application accessible.

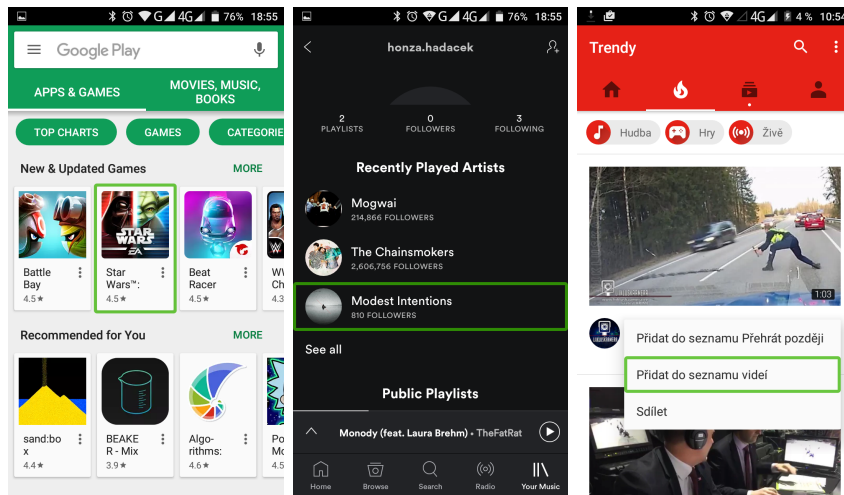
Like the *Core System*, TalkBack uses gestures for navigation. It is, however, way more complex to use. It uses a virtual cursor which designates the UI element which is currently active. This cursor can either be linearly moved over the component tree (in a heuristic order) by gestures (similar to the *Core System*) or can be “dragged” by user’s finger while the description of what’s under the cursor is announced (so-called “explore by touch” mode).

TalkBack uses substantially more gestures than the *Core System* for navigation:

- slide left to right - move cursor forward
- slide right to left - move cursor backward
- slide top to bottom - increase reading granularity (characters - words - lines - paragraphs)
- slide bottom to top - decrease reading granularity
- double tap - activate current item
- dragging top-down + left-right - global context menu
- dragging down-top + left-right - local context menu
- dragging top-down + right-left - go back
- dragging down-top + right-left - main screen
- dragging right-left + bottom-top - last applications
- drawing V on the screen - go to the last item in the list
- drawing an upside-down V on the screen - go to the first item in the list
- dragging two fingers from top edge of the screen to bottom - display Android notification shade
- dragging two fingers from the middle of the screen to bottom or top - scroll up or down (text, list, ...)
- tap once and hold - perform long press on current item

The global context menu contains more options like (“read all text”, “spell the last utterance” or “text-to-speech settings”). The local context menu items can be populated

## 2. The “Core System” and other mobile accessibility solutions



**Figure 5.** Examples of graphically-oriented user interface: Google Play, Spotify, Youtube

by the application to provide shortcuts relevant to accessibility (i.e. navigation by headings in a web browser).

TalkBack uses the standard Android keyboard. Default Android keyboards have support for accessibility and will announce letters currently below user’s finger. The letter is input by releasing the finger.

TalkBack only focuses on application accessibility. It does not provide any assistive technology (like OCR - text recognition), nor it provides a text-to-speech service. These have to be installed separately. TalkBack comes with a tutorial that explains the gestures and allows a new user to practice the operation of their phone.

While the best solution for making a native application accessible, it has several serious drawbacks:

1. even though TalkBack should make all applications accessible, many applications are not usable due to sloppiness of their developers (lack of text labels, custom widgets and gestures) or due to their nature (e.g. most games)
2. users find the system of gestures too complex to learn and remember
3. most modern applications are visually oriented and use layouts that are complex and change during their use, which makes non-visual orientation extremely challenging (see Figure 5)
4. standard keyboard with voice feedback is not very convenient to use, especially for beginners

## 2.3. Complete solutions

### 2.3.1. Kapsys SmartVision2

SmartVision2 is a commercial solution by a company called Kapsys[11], which is built upon Android 6 and is being sold on dedicated devices. Unlike the *Core System*, it works with the standard Android interface. It has a physical keyboard of classic design, and its control philosophy builds upon TalkBack. The presence of the physical keyboard, however, simplifies navigation and control of Android applications to some extent.



As for Accessibility tools, it comes with Color detector, Light detector, digital Magnifier and an off-line photo OCR (which unfortunately seemed to be defunct on a sample borrowed for the purpose of this thesis). It also offers NFC interface for daily objects tagging and recognition.

### 2.3.2. Claria Vox

Claria Vox was another commercial solution developed by a company called Claria[12]. During the writing of this thesis, the company went out of business. Claria Vox attempted to simulate a physical keyboard with a grid rubber overlay placed over a standard touchscreen. It had a custom user interface with support for Android applications vocalized by standard TalkBack. Unlike other solutions controlled by gestures, Claria Vox has a menu with numbered options. To select an option, one presses a corresponding number on the keyboard grid.

Claria Vox came with some vision aids; Camera and gallery (with face detection and photo tagging as an assistive feature), off-line OCR (German, French, English, Spanish, Italian and Dutch), barcode scanner with product search and illumination level detection. A version of Claria for low-vision users, called Claria Zoom, can be downloaded free-of-charge on Google Play. While using a different interface, it contains the same vision aids present in Claria Vox. A brief test done as a part of this work suggest that the OCR function in Claria fails to handle with reasonable accuracy even relatively simple scenes presented in Chapter 5.

### 2.3.3. Corvus

Corvus[13] is a self-proclaimed “accessible kit” (sic!) for Android developed by a Slovak company Stopka. It is an application that can be purchased and installed on any phone with Android. Phones with pre-installed Corvus can also be purchased. A demo version limited for 10 minutes is available from the company website. Corvus has a custom, gesture-operated user interfaces with high-contrast text and graphics. It also has a screen-reader that integrates to Android similarly to TalkBack, using its custom gestures.

The philosophy of Corvus is rather similar to the *Core System*, but the system of gestures is significantly more complex. Accessibility tools in Corvus include a light detector and QR code reader, which is intended for tagging of daily objects.

## 2.4. Specialized applications

### 2.4.1. TapTapSee

TapTapSee[14] by CamFind Inc. is probably the best known assistive application for object recognition. It uses a combination of object recognition, machine translation, photo-OCR and human annotators to extract information from an image. Originally, the application would only allow for limited number of recognition attempts and then require user to buy credits. As of 2017, this limitation seems to be lifted and the



**Figure 6.** Commercial Android accessibility solutions. Left to right: Kapsys SmartVision2, Claria Vox and Corvus

application can be used free of charge, without any explicitly stated limits (there is probably still some kind of fair-user-policy on the number of recognitions performed).

TapTapSee is an on-line solution and cannot work without Internet connection. Main drawback of this application is that the recognition is very slow. In some cases, it may take more than 30 seconds.

#### 2.4.2. KNFB Reader

KNFB Reader is a commercial OCR application by Sensotec[15]. It is primarily intended as an assistive tool (accessibility of the application interface itself is mediated by the TalkBack service). It uses a well-know state-of-the-art OCR engine developed by a Russian company called ABBY[59]. The application is offered free-of-charge, but the number of recognitions is limited to 25 pages in the trial mode.

KNFB Reader is mostly intended to handle classical paper documents or books. However, it can handle some more complicated scenarios too. The most important advantage of this solution is that it is able to work without Internet connection. The most important drawback is the price for full version which, as of 2017, was about 100 Euro.

#### 2.4.3. Darwin Wallet

Darwin Wallet is an open-source (GPL) banknote recognition application for Android, primarily intended as an assistive tool. Banknote recognition is very important in case of currencies where values cannot be distinguished using banknote dimensions, like US



**Figure 7.** Examples of specialized accessibility tools: TapTapSee, KNFB Reader and Darwin Wallet

dollars.

Darwin Wallet supports US and Australian dollars, British pounds, Euros, Russian and Belarusian rubles, Georgian laris and Ukrainian hryvnias.

From the technical point, it uses OpenCV[16] library, Oriented BRIEF[17] detector/descriptor and K-NN based tentative matching. A (potentially) banknote image is classified according to maximal relative number of matches to each of the templates in the database. There is no feature weighting or geometry verification. There is only one sample of each of the banknotes. The algorithm is reminiscent of implementation introduced in [7], but it is written in native C++ instead of Java, which improves real-time performance.

## 3. Specification of Computer Vision tasks relevant for accessibility

In the previous chapter, we have discussed few existing applications and solutions, some of them incorporating computer vision algorithms. This chapter discusses possible uses of general computer vision algorithms and techniques in assistive features.

### 3.1. Object recognition

The object recognition problem has been one of the central problems in the Computer Vision field. The goal of an object recognition algorithm is to distinguish and identify objects in images or video sequences. While humans can recognize objects in images with very little effort, in most cases at least, this task is one of the hardest ones in the computer vision field. The reason for this task to be problematic is that a complex three-dimensional object can look very different when projected on a two-dimensional plane from different angles or distances. In the real world, objects are often partially occluded, and there is a multitude of objects present in a single image.

While very complex, general object recognition only has limited applicability as assistive technology. Most objects of daily life can be more easily and reliably distinguished by touch or other non-visual clues when physically available to a visually impaired person. General object or scene recognition algorithm can, however, be a useful tool when a blind person is faced with the task of identifying an unknown digital image.

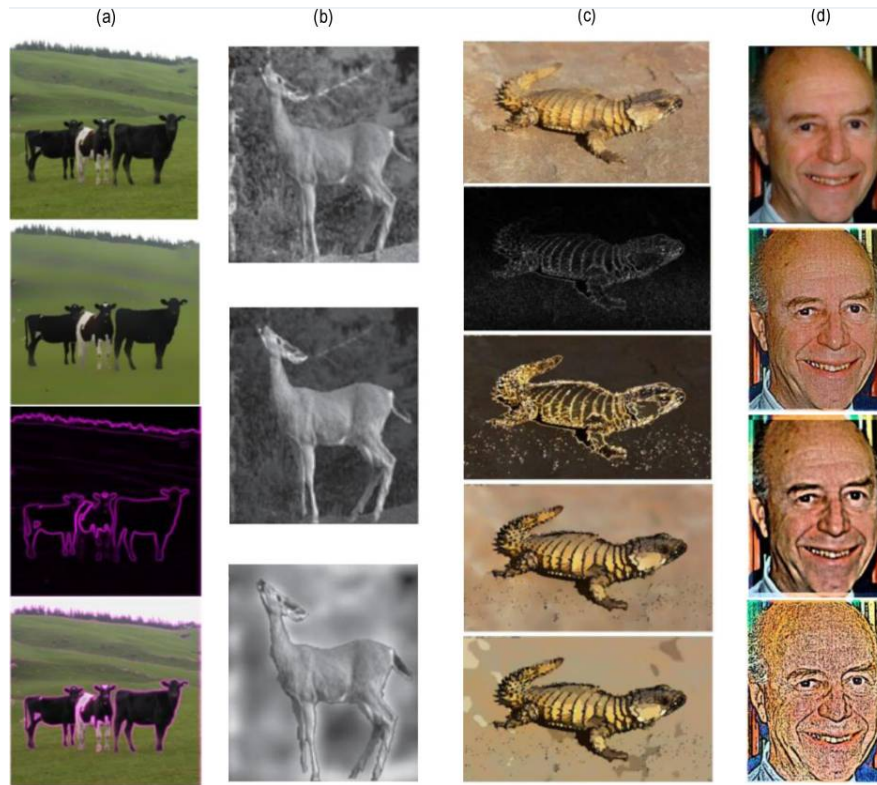
There are three basic sub-problems of object detection and recognition, sorted by their difficulty:

1. segmentation and object detection
2. single class object recognition
3. multi-class object recognition

#### 3.1.1. Segmentation and Object Detection

The goal of segmentation is to partition an image into multiple (usually either disjoint or hierarchical) sets of pixels.[18] The reason why segmentation is done is to simplify the image for further processing. There are many different approaches to segmentation, only the most popular will be mentioned in this thesis:

- segmentation by pixel color alone (local and global thresholding, k-means segmentation)
- segmentation by pixel color inter-pixel measures (Huttenlocher-Felzenszwalb method [19], watershed algorithm)
- using edge detection (Canny edge detector [20], some heuristics to produce closed edges can be if needed)



**Figure 8.** Examples of segmentation and other filters for “image enhancement” intended for low-vision users.[24]

- using a convolutional neural network [21]
- semi-automatic segmentation (Grab-cut [22], SIOX [23])

A good application of segmentation related to accessibility is a low-vision filter (image “enhancement”). An overview of image processing methods used for low-vision is available in [24]. A particular implementation is discussed in [25]. Figure 8 shows examples of such image filters.

Segmentation can be used to partition and simplify the scene and enhance the mutual contrast between segments to improve perception by people with severe vision loss. Simpler methods like edge detection seem to give quite good results in this particular tasks as the segmentation need not be perfect for human interpretation.

### 3.1.2. Single class object recognition

The goal of single class object recognition is to recognize a single dominant object present in the image. As with segmentation, there are multiple approaches. The limitation of single class object recognition is that the accuracy of recognition can be negatively impacted by other objects present in the image (in some cases, it is even clear which is the primary object).

There are three basic approaches to single class object recognition:

#### Template based methods

These methods are based on a similarity between an example image from a database (having a known class) and image patch extracted from the image being recognized (possibly by sliding window technique).

*Greyscale matching* and *gradient matching* are the simplest methods that use cross-correlation measure on greyscale or gradient images to decide whether a window cut out of a picture matches the template or not.

More sophisticated methods are eigenfaces (PCA) and fisherfaces (LDA), which are popular for face recognition.

The most important advantage of template based methods is their holistic approach and robustness to illumination changes, local occlusions and defects. The disadvantage is that that methods based on templates are not very robust to deformations.

#### Feature based methods

These methods are based on *features* that are apriori extracted from the image being classified. Each feature is represented by a point within the image and some descriptor, which is usually a float or binary high dimensionality vector. The trained database consists of a list of key points and their descriptors, which is kept for every image in the database.

A simple pipeline for feature a based method is depicted on Figure 9.

For larger databases, this basic pipeline is usually modified (mostly for performance but also for other reasons, like feature weighting) and extended to adopt so-called Bag-of-words model.

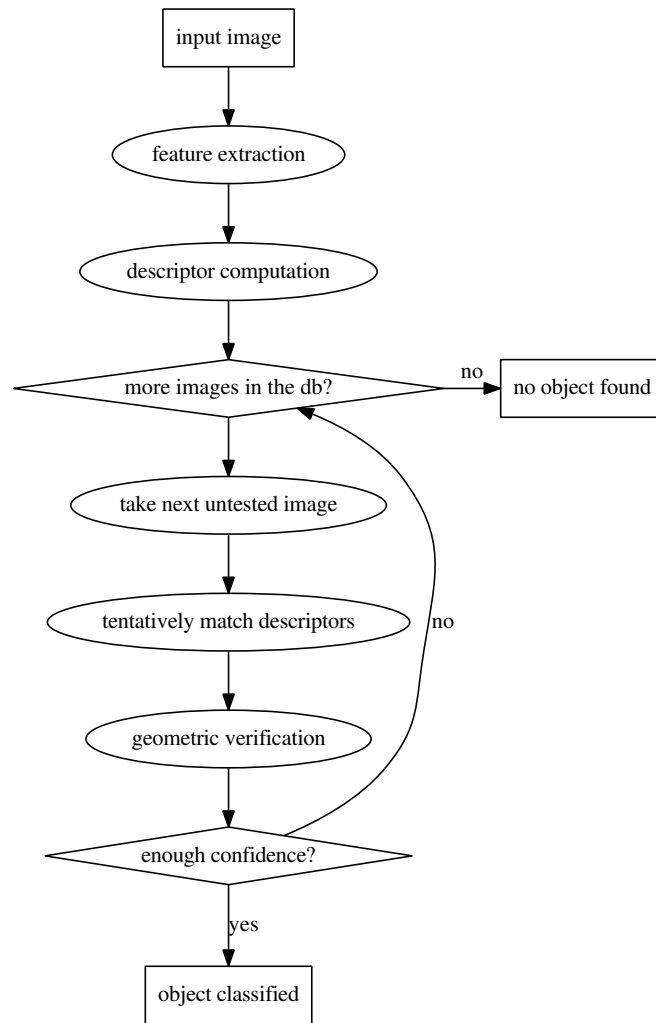
#### Methods based on convolutional neural networks

Single class image classification was one of the first problems successfully tackled by convolutional neural networks. In particular, the Caffe framework[26] is one of the very successful libraries pioneering single-class recognition by a neural architecture in the ImageNet Large Scale Visual Recognition Challenge.[27] It is an open-source solution, including the training parameters and the pre-trained network for image recognition. The Figure 10 shows an example of recognition results produced by Caffe.


#### 3.1.3. Multi-class object recognition

The goal of multi-class object recognition is to extract more complex description from the image. The task is arguably much harder than single-class object recognition, but the results are more useful in the context of assistive technology. The usual goal is to obtain a caption describing the image in plain words a human annotator might have written.

One of the recently published state-of-the-art methods for multi-class object recognition is “Show, Attend and Tell”[28]. It uses so-called “attention” based framework to focus



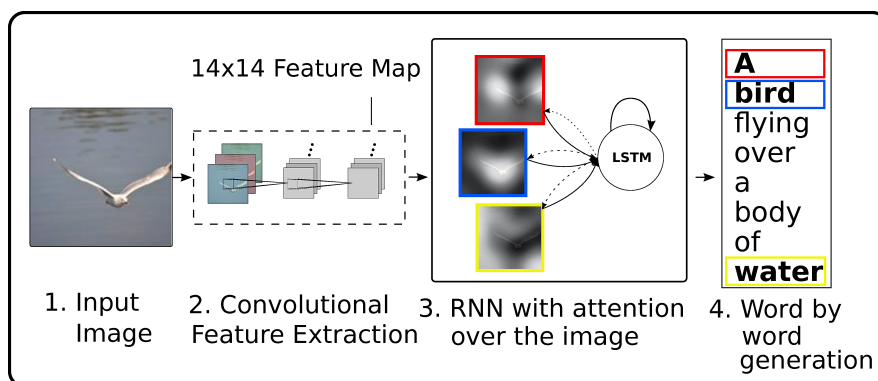
**Figure 9.** A flow-chart of a simple feature based single class recognition method.



Maximally accurate	Maximally specific
dog	1.21359
hunting dog	1.17208
domestic animal	1.10901
canine	1.08033
carnivore	0.86467

CNN took 0.068 seconds.

**Figure 10.** Single class object recognition: Caffe[26] on-line demo classification results of a detailed image of the dog of the author of this thesis. It's remarkable that Caffe was able to give a sensible classification even though the image is not a very typical image of a dog.



**Figure 11.** An infographic from [28] illustrating the basics of so called “attention” framework for multi-class recognition.

on important parts of the image. Much in the same way human vision fixates when you perceive a scene, the model learns to “attend” to selective regions while generating a description. This “attention” part is done using recurrent neural network and a word generator generates the final caption.

### 3.2. OCR / Text-in-the-wild recognition[29]

The problem of text-in-the-wild recognition, also known as photo-OCR or real-scene text recognition, is a challenging problem of computer vision. There are specific issues in text-in-the-wild recognition that make it more difficult than recognition of printed documents, which is considered to be a virtually solved problem.

The application of OCR for mobile accessibility is difficulty-wise somewhere between traditional document OCR and full-blown unconstrained text-in-the-wild recognition. While a mobile camera shot of a daily object or document with texts to be recognized is generally simpler than complex images of texts present in the street or similar environment, it is still significantly more challenging than a traditional document scanned under controlled conditions.

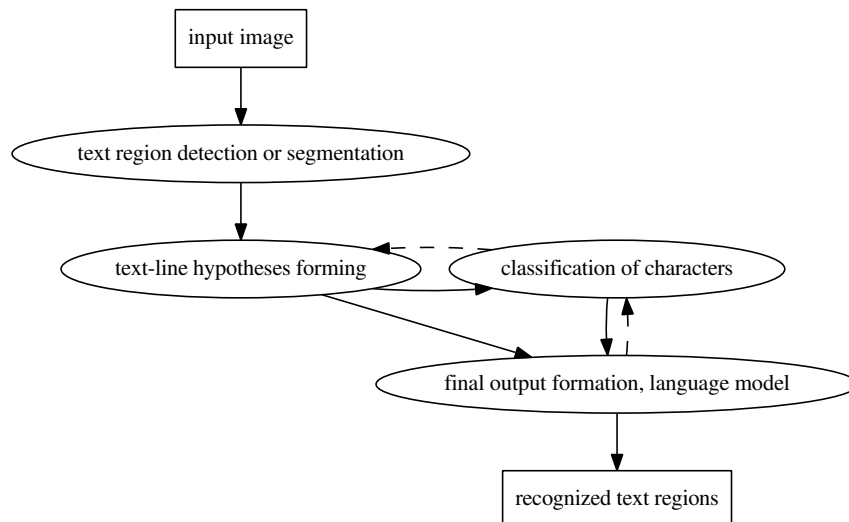
Common typographical conventions present in printed documents are often ignored in product labels and less formal “documents” which complicates the process of text-line estimation and decision about word boundaries. Geometric distortions caused by a camera photographing labels from various angles are present, as well as distortions introduced by the camera lens itself. A real-scene photograph features various illumination conditions, which furthermore complicates the matter.

Traditionally, a photo-OCR system is a cascade or “pipeline” of several components. Each component performs an isolated part of the task. There may be feedback loops feeding back the information obtained at later stages to the previous stages to further refine precision of the system.

#### 3.2.1. Algorithms based on image binarization

The key advantage of a method based on image binarization is that scale invariance is easy to achieve because textual regions of all scales and rotations can be extracted





**Figure 12.** A traditional OCR pipeline. Most successful solutions have feedback loops (dashed) that allow refinement of previous steps in the cascade based on the succeeding steps (i.e. refinement of text-line hypotheses based on character classification).

from an image in a single pass. The main disadvantage is that this process produces a set of connected components (blobs) and ignores any relations between these blobs completely.

The binarization technique in this context is usually some kind of thresholding, either global or local, or more sophisticated thresholding methods (which need not to produce disjoint results), like MSER [18].

In practice, it means that if a glyph consists of more blobs, it is necessary to relate them somehow, and if any of them can get lost during the process of binarization, it is impossible to recover it without a complementary method. The method is dependent on the fact that most of the letter regions consist of areas having uniform color (single-color ink) and may have problems with textured letters or backgrounds.

A good example of an algorithm based on image binarization, in particular, an MSER modification called CSER, is the TextSpotter algorithm developed by Matas and Neumann from the Czech Technical University [30].

### 3.2.2. Algorithms based on edge detection

Methods exploiting edge detection are based on the assumption that a relatively steep image gradient is present on borders of letters and numerals. Such gradient can be detected by a standard edge detector (e.g. Canny). The most prominent method utilizing the edge detection is the Stroke Width Transform described in [31]. The final output of this method is a set of connected components, similar to methods based on image binarization, however, the blobs are not obtained directly. A sophisticated algorithm utilizing the notion of Stroke Width consistency is used to transform the edges into component components.

This method is more robust to textured ink and background than image binarization. It performs poorly on blurred glyphs because they are not featuring significant edges to

### 3. Specification of Computer Vision tasks relevant for accessibility

separate a glyph from its background. Also, the SWT may fail in some outlined fonts because it has difficulties to distinguish between a glyph itself and its outline.

#### 3.2.3. Sliding window approach

Sliding windows have been utilized in several works, e.g. [32]. Sliding window detectors have an advantage of being potentially more robust in situations that involve connectivity defects than the methods mentioned above.

Letters may contain defects resulting that an originally single-component glyph can be split into multiple connected components. Such letters will not be extracted by an image binarization technique as a single object (unless some explicit post-processing technique to merge such letters is used). The technique of sliding windows can directly extract multi-component letters, which simplifies their further processing.

#### 3.2.4. Algorithms based on deep neural networks

According to ICDAR 2015 Robust Reading Competition[33], which is one of the best available benchmarks to compare photo-OCR methods, the most successful modern photo-OCR methods are based on neural architectures [34] [35] [36].

These methods differ in exact details, and detailed description is beyond the scope of this thesis. Some of them use traditional approaches like binarization as the first step for text detection (e.g. the method **TextRecognizer** more described in section 5.1.1 seems to be doing it to improve real-time performance).

Some of them are fully neural architectures, end-to-end, these tend to be much slower, but more robust to multiple connected component problem mentioned above. These architectures also require arguably less hand-tuning, given enough annotated data suitable for training is available (which may be a great obstacle for smaller research teams).

## 4. Banknote recognizer implementation

As a part of this thesis, an algorithm for banknote recognition has been developed. The goal of this algorithm is to recognize values of paper banknotes presented to the phone camera. The procedure should be robust, and training of new banknote sets should be a quick and easy task not requiring the developer to be an expert in machine learning.

### 4.1. Functional requirements

Let's define formal requirements for the banknote recognition solution:

- reliability, especially low number of false classifications
- off-line operation
- compactness of the database
- low runtime speed and memory consumption
- high robustness to lighting conditions

The algorithm has the following input and output:

**Input:** A sequence of images from the phone camera that might contain a banknote.

**Output:** Value of the recognized banknote or rejection (= don't know).

The algorithm itself is presented to a user as a continuous camera capturing activity, which is interrupted once a banknote is detected and recognized.

Conceptually, a modification of a traditional computer vision feature-based approach described in 3.1.2 was used.

### 4.2. Dataset

The dataset for one currency consists of a complete set of scanned banknotes, both front and back sides are used. Since it should be possible to add new datasets easily, only a single exemplary of each banknote is used. To address the problem of proper classifier training, a data augmentation technique is used. The goal is to synthesize more examples of each banknote.

### 4.3. Pre-processing and data augmentation

The original input data are high-quality flatbed scanner images of banknotes, properly cropped with serial numbers blurred. Then, they are downscaled to 750 px width. Such examples are ideal compared to real-life banknote snapshots. Therefore a richer training material to better represent real-life examples was needed.

#### 4. Banknote recognizer implementation



**Figure 13.** Original 10 Euro banknote scan and 5 synthesized views selected from the dataset used for training.

Before the training, the data set is augmented by applying a random homography to the image.

$$\mathbf{H}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$$

Where  $R_{ij}$  are random floats from range  $(-0.01, 0.01)$ .

Random translations in  $x$  and  $y$  are then added:

$$t_x = \frac{w}{2} + q_w$$

$$t_y = \frac{h}{2} + q_h$$

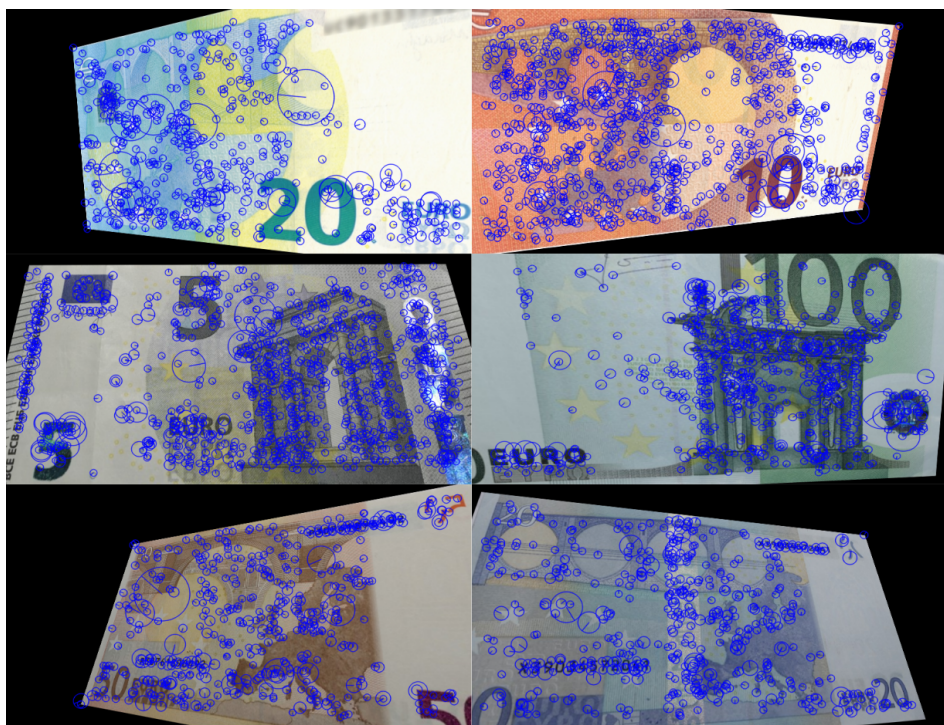
$$\mathbf{H} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{H}_0 \cdot \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix  $\mathbf{H}$  is the final homography used to transform the image (OpenCV implementation `warpPerspective`) is used. Image width and image height in pixels are denoted by  $w$  and  $h$  respectively. Numbers  $q_w$  and  $q_h$  are random floats from interval  $(-150, 150)$ . Figure 13 shows the original 10 Euro banknote scan and few augmented training samples.

For each banknote, 40 different views were synthesized, 20 for each side. For Euro banknotes, this gives the dataset size of total  $8 \times 40 = 320$  training images (values 5, 10, 20, 50 and 100 were used; 10 and 20 and 50 comes with two graphically different versions).

#### 4.4. Keypoint detection and description

After some experiments, the BRISK[37] algorithm was chosen as the best balance between computation and storage requirements of the descriptors vs. robustness and



**Figure 14.** Detected keypoints on few training samples.

repeatability. Some experiments with SURF[38] algorithm were made, however, high storage and CPU demands out-weighted the reliability improvements, that were relatively negligible in practice.

Another advantage of BRISK is that it is a patent-free algorithm, so a software containing this algorithm can be freely used even in countries where software patents are recognized (e.g. The United States).

After some experimentation, the best parameters for BRISK detection were determined to be  $t = 10$ ,  $n = 5$ ,  $p = 1$  ( $t \sim$  threshold,  $n \sim$  number of octaves and  $p \sim$  pattern scale).

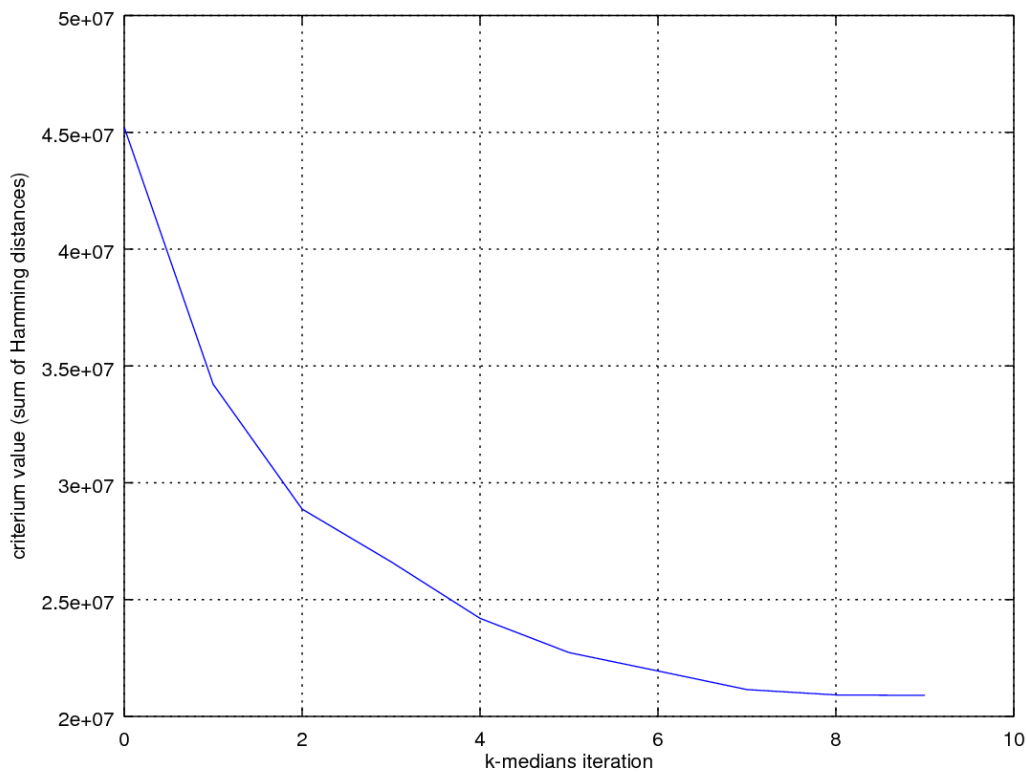
The neighborhood around each of the key points is described using a standard BRISK descriptor, which is a binary (512-bit long vector) descriptor. The similarity between two descriptors is computed simply as a Hamming distance.

## 4.5. Bag of Words

After performing data augmentation and computing the keypoints and descriptors for each of the training images, we end up with a database of 320 images, together containing about 200 000 key points, each key point having a 512-bit long feature vector. This is a relatively large amount of data for processing on a mobile phone, especially when near real-time processing is the goal.

Since it would be computationally prohibitive to implement exhaustive tentative matching between an image to be classified and all images in the dataset, a bag-of-words approach was chosen. As a desirable side-effect, this approach also helps to distinguish

#### 4. Banknote recognizer implementation



**Figure 15.** The graph of the change of k-medians error dependent on iteration (sum of Hamming distances from respective centroids)

meaningful features from meaningless also (since we are doing classification here and banknotes generally contain some similar graphical elements).

The problem with binary descriptors in the context of the Bag of Words technique is that (unlike for real-valued vectors) no well-established technique exists to learn the codebook from the training database of binary vectors. This is not an easy task because the problem of finding a median string is NP-complete[39].

After some investigation, a generalized approximate string median algorithm[40], namely the implementation presented in [41] was used. Although a more time-efficient algorithm for approximate binary string median most likely exists, the codebook is only computed once and ahead of time (and a standard computer can be used), usage of a general string median algorithm is acceptable for this purpose.

The codebook was then computed using a k-medians algorithm with the k-means++ initialization[42] that was customized to work with Hamming distance instead of the standard L2 distance. The final number of codewords used was 400, which was determined experimentally (a higher number of codewords did not seem to change the final accuracy very much). The progress of error during algorithm iterations is shown on Figure 15.

## 4.6. Classification

After the codewords are obtained, the whole database is encoded using those codewords. Each training image  $i$  is then represented as an integer-valued vector  $\mathbf{x}_i$  of length 400 (number of codewords) and the class label  $y_i$ .

The classifier of choice for this problem was a Gradient Boosted Tree Forrest[43] model.

Similar to AdaBoost[44], this classifier iteratively builds a strong classifier from an ensemble of weak ones, minimizing an internal error estimate (loss function). In this case, the weak classifiers are fixed-height decision trees. The main advantage of this kind of classifier is that it estimates a metric internally, so it does not depend on the variable scale like other popular methods. Since we expect the variables to vary a lot in their discriminative power, the problem of scaling and metric is something important to avoid.

Some other classifiers, like Random Forests [45], SVM[46] and Multi-class Adaboost[47] have also been briefly tested, but Gradient Boosted Trees proved to be the most successful and relatively easy to train. The implementation `GBTrees` from OpenCV[16] was used. The classic TF-IDF heuristic[48] was also tried, but the accuracy was relatively low.

The Gradient Boosted Trees Forest has the following parameters:

- shrinkage (regularization parameter) - the value 0.08 was chosen using 5-fold cross-validation, see Figure 16
- number of trees - the chosen value was 300 (the more trees, the more expressive and the more likely overfit model will be produced)
- loss function - for a classification problem, the standard loss function to use is cross-entropy (deviance) loss.  $K$  such functions are build, one function for each output class. The final function is

$$L(y, f_1(x), \dots, f_K(x)) = - \sum_{k=0}^K 1(y = k) \ln p_k(x)$$

where

$$p_k(x) = \frac{\exp f_k(x)}{\sum_{i=1}^K \exp f_i(x)}$$

is the estimation of the probability of  $y = k$ [16].

- maximum depth of a single weak classifier tree - the default value 3 was used here

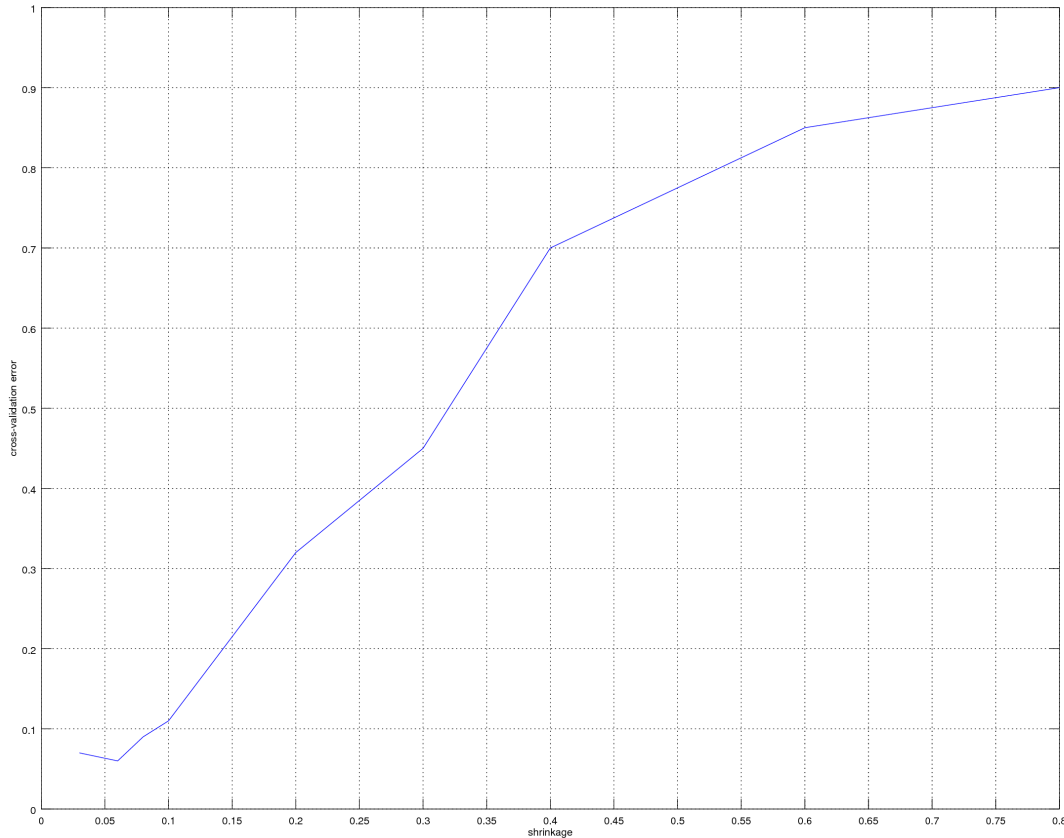
The final error of the classifier on real-world data taken by the phone camera was 7.8%.

## 4.7. Geometric verification

To further improve resilience to false classification and to implement rejection decision, another step in the algorithm was added - geometric verification, which is done via RANSAC[49] homography estimation.

From the classification step, we have the most likely class. Since each class corresponds to a single image scan, it is possible to estimate a homography mapping between the camera image and banknote prototype.

#### 4. Banknote recognizer implementation



**Figure 16.** The graph of cross-validation error dependent on shrinkage parameter.

We assume that a banknote is more or less a plane and we want to find a homography that maps key points from the camera image on the key points of the template image.

We are looking for a homography matrix  $\mathbf{H}$  between inlier points in the camera image  $(x'_i, y'_i)$  and the points in the corresponding candidate template identified by the classifier  $(x_i, y_i)$ :

$$u_i \cdot \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \sim \mathbf{H} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

so that the back-projection error is minimal[16]:

$$\operatorname{argmin}_H \sum_{i \in \text{inliers}} \left( x'_i - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left( y'_i - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2$$

For homography computation, tentative correspondences are needed. These are found using the simple “brute force” matcher implemented in OpenCV. As only a single matching between two images is done, this is acceptable.

The threshold for RANSAC outliers is chosen to be 20 pixels, which would be extremely permissive in other CV tasks, but here we expect (but don’t explicitly correct) lens distortion slight bending of the banknote itself, so this high threshold was chosen to accommodate for this.



The RANSAC algorithm provides us with the matrix  $\mathbf{H}$  and a list of key point inliers. The final decision (whether we indeed see a banknote or not) is made using few heuristics:

1. Some **absolute number of tentative correspondences** is always needed to compute homography reliably. We require **at least 20** here. If less is present, we do not even attempt RANSAC and the image is rejected as non-banknote.
2. Given the hypothetical  $\mathbf{H}$  returned by RANSAC, we can compute the **relative area** of the picture the banknote occupies. The banknote must not be extremely small or extremely close. Only banknotes that occupy **between 20% and 150% of the image** are considered plausible. Results outside of this range are rejected as non-banknotes.
3. The **absolute number of inliers** must not be too low. If it is borderline (**less than 8 points**, where 4 points are the absolute minimum for homography), the image is rejected as non-banknote.
4. The **relative number of inliers** must not be too low. We require that **at least 10% of the tentative correspondences be inliers**. Else the image is rejected as non-banknote.

## 4.8. Results

The algorithm was tested on a dataset consisting of real pictures of Euro banknotes taken by the camera of the *Core System*.

The testing dataset contained the following classes 5A0, 5B0, 10A0, 10A1, 10B0, 10B1, 20A0, 20A1, 20B0, 20B1, 50A0, 50A1, 50B0, 50B1, 100A0, 100B0, nothing. Each banknote class was included five times, in different angle and lighting conditions. The banknotes used for testing were physically different from those used for training. 10 images of non-banknotes were also included in the dataset. This results in total 90 testing banknotes.

correctly classified banknote	86.25%	68
correctly rejected non-banknote	100%	10
incorrectly rejected banknote	13.3%	11
incorrectly classified banknote	1.1%	1
total	100%	90

## 5. OCR implementation

This chapter describes the implementation of the text recognition feature for the *Core System*.

### 5.1. OCR engines

Since implementation of a high quality OCR engine is beyond the scope of this thesis, two off-the shelf photo OCR engines were selected: *TextRecognizer* from *Google Play Services* and a OCR cloud service, which is a part of *Microsoft Cognitive Services*. An attempt was made to combine the strengths and weaknesses of those two off-the-shelf solution.

The goal of an OCR engine is to take an image from the camera and return a list of labeled bounding boxes that correspond to detected text hierarchy (words / lines / paragraphs). The bounding boxes should preferably contain some confidence measure to give some hint for further processing.

#### 5.1.1. Google TextRecognizer

Google TextRecognizer is an off-line OCR engine for Android operating system. It's designed to give almost real-time performance while maintaining reasonable accuracy. The original algorithm behind TextRecognizer was developed by a company called Quest Visual and made public in their Android application called WordLens[50], which provided an augmented reality translator. Quest Visual was later acquired by Google, who made the algorithm available *free of charge* on every device that has Google Play Services installed, which includes the *Core System*.

The exact algorithms behind TextRecognizer are trade secret, but an informed guess can be made about the choice of internal algorithms.

A brief look at the disassembly of the binary `libocr.so`, which is included with Google Play Services package and is responsible for the OCR, suggests that the image is segmented by some kind of binarization algorithm (namely the adaptive Sauvola and global Otsu methods seem to be used) prior as the first stage of the OCR pipeline. It was observed that only a greyscale image is passed to the native library through the JNI. The presence of functions `pixOtsuAdaptiveThreshold` and `pixSauvolaBinarize` from the Leptonica library[51] gives more hints about segmentation algorithms possibly used. This finding is confirmed by observation of the algorithm behavior, namely its almost real-time speed and inability to classify letters that cannot be extracted as a single connected component in greyscale.

Segmented regions are most likely classified using a deep net powered by the TensorFlow library[52], also developed by Google (`ocr::photo::MagnetCharClassifier` in

the library disassembly). The final text is produced by an n-gram language model. The library explicitly supports Danish, Dutch, English, Finnish, French, German, Hungarian, Italian, Latin, Norwegian, Polish, Portuguese, Romanian, Spanish and Swedish. It however does not seem to use an explicit dictionary and the n-gram model it uses is able to give reasonable results for other languages (including Czech language).

The main disadvantage of TextRecognizer is lower accuracy and lack of any programmer-tunable configuration that would allow to customize the recognizer behavior for better fit with intended application. Usage of TextRecognizer is demonstrated in Snippet 1.

**Snippet 1.** A basic function to recognize text inside a Bitmap using TextRecognizer

```
String readText(Bitmap input) {
    TextRecognizer offlineOcr = new TextRecognizer.Builder(getContext())
        .build();
    SparseArray<TextBlock> result = offlineOcr.detect(new Frame.Builder()
        .setBitmap(image).build());

    String recognitionResult = "";
    for (int i = 0; i < result.size(); ++i) {
        TextBlock item = result.valueAt(i);

        List<Line> lines = (List<Line>) item.getComponents();
        for (Line line: lines) {
            recognitionResult += line.getValue() + "\n";
        }
        recognitionResult += "\n";
    }

    return recognitionResult;
}
```

### 5.1.2. Microsoft Cognitive - OCR

Microsoft Cognitive Services [53] is an umbrella name for several computer vision and machine learning cloud services provided by Microsoft. All of these services are accessible using a relatively simple REST API[54] with JSON[55] output. It is a paid service, however, Microsoft offers free licenses which are limited in maximum monthly number of requests. The limit of the OCR service is 5000 requests monthly, which is more than enough for evaluation purposes.

This OCR service supports the following languages: Mandarin, Czech, Danish, Dutch, English, Finnish, French, German, Greek, Hungarian, Italian, Japanese, Korean, Norwegian, Polish, Portuguese, Russian, Spanish, Swedish and Turkish.

Communication with the service is achieved using a simple HTTP POST request. The body of the request can either contain an URL of an image that is downloadable on-line, or binary data representing an image in JPEG, PNG, GIF or BMP formats.

If the recognition procedure is successful, a JSON encoded response is returned. A sample of this response can be seen in Snippet 2.

**Snippet 2.** A sample recognition result returned by Microsoft Cognitive

```
{ "language": "en",
  "textAngle": 0,
  "orientation": "Up",
  "regions": [
    { "boundingBox": "439,317,922,219",
      "lines": [
        { "boundingBox": "439,317,922,91",
          "words": [
            { "boundingBox": "439,332,287,67",
              "text": "IAESTE" },
            { "boundingBox": "757,324,228,70",
              "text": "Czech" },
            { "boundingBox": "1019,317,342,91",
              "text": "Republic" }
          ]
        },
        { "boundingBox": "485,435,827,101",
          "words": [
            { "boundingBox": "485,452,122,68",
              "text": "Let" },
            { "boundingBox": "633,466,86,52",
              "text": "us" },
            { "boundingBox": "751,442,185,94",
              "text": "light" },
            { "boundingBox": "953,458,199,76",
              "text": "your" },
            { "boundingBox": "1178,435,134,89",
              "text": "fire" }
          ]
        }
      ]
    }
  ]
}
```

The request to remote server is made using the Android Volley[56] library and the result is decoded using Google GSON[57]. The principles behind this OCR solution are a trade secret of Microsoft. It does not seem to use thresholding of any kind, at least not exclusively.

### 5.1.3. Comparison - summary

The following table shows most important qualitative differences between the two engines:

solution	TextRecognizer	Microsoft Cognitive
usual recognition time	cca 3 seconds	cca 15 seconds
off-line	yes	no
pricing	free	paid per request
language support	basic n-gram model	full model for 20 languages
characters must be CC <sup>1</sup>	yes	no
practical accuracy	mid-high	very high

To get some idea about the quality of used OCR engines, both of them were evaluated on ICDAR 2015 Robust Reading competition[33], Focused Scene Text database. It is

important to notice that typical ICDAR scene is different from a typical scene expected for this application.

The following results were produced by the Deteval[58] protocol on ICDAR 2015 dataset (full-resolution ICDAR images used):

	precision	recall	h-mean	avg time
TextRecognizer	0.661	0.767	0.680	4.2s / image
Microsoft Cognitive	0.832	0.908	0.868	13.3s / image

## 5.2. OCR user interface

Having two relatively strong OCR engines at hands, the main problem of this task was how to design the user interface that will be easy to use even by people with no useful vision.

Since TextRecognizer is giving almost-realtime results, a decision was made to use it to perform live recognition on the camera feed. Microsoft Cognitive gives much better result, but can only be reasonably used in a single-shot fashion, it is not suitable for continuous recognition.

Considering all circumstances, the following application workflow has been devised:

1. camera/document positioning mode with immediate (and inaccurate) recognition results
2. text reading mode, featuring browsing of the text by paragraphs and additional options (save as note, send as text)

### 5.2.1. Camera/document positioning mode

The purpose of this mode is to allow quick examination of the document, with possibly limited accuracy, but immediate results. Those results are continuously being announced - all text detected in the image is linearized and enqueued for text-to-speech. Use can reset the TTS queue any time by tapping the screen.

#### Confidence score

The results TextRecognizer returns have a confidence score (a float number 0-1) assigned to them. Those are not directly available by Google API, but they can be unofficially retrieved using reflection. This confidence score is very useful since sometimes the text can be detected but not recognized for various reasons. We can employ several heuristics that are based on this confidence score to improve user experience.

To distinguish “confident” results from “gibberish” results, the following rule of thumb is used: **A “Line” is considered confident, *iff* the minimal confidence of all words within this line is no less than 0.6.** This rule is used to label each “Line” as confident or inconfident. Only confident lines are enqueued for text-to-speech.

### Upside-down detection

The TextRecognizer is reasonably robust to rotation, but will not recognize texts that are upside down. Those text would be detected, but no sensible text is recognized. This situation can be detected user simple heuristic:

1. if less than 3 lines are detected, don't attempt to detect the upside-down situation
2. if more than 3 lines are detected, compute the percentage of confident lines. If it's less than 0.5, than we inform the user that we have detected text which is not readable and they should attempt to rotate the page/object by 180 degrees.

### 5.2.2. Text reading mode

The purpose of the text reading mode is to allow user take a snapshot of current document and comfortably examine the detected text. A still picture of the document is taken and an attempt is made to recognize the content of the picture using Microsoft Cognitive OCR.

The photograph is scaled down to 1600x1200 pixels and compressed by JPEG at quality=70. The result is sent via the REST API to the remote server for OCR. If the connection fails or if the result is not delivered within a 30 second time-out, a fall-back method of recognition is used. The fall-back method consists of feeding a full resolution picture to TextRecognizer for recognition.

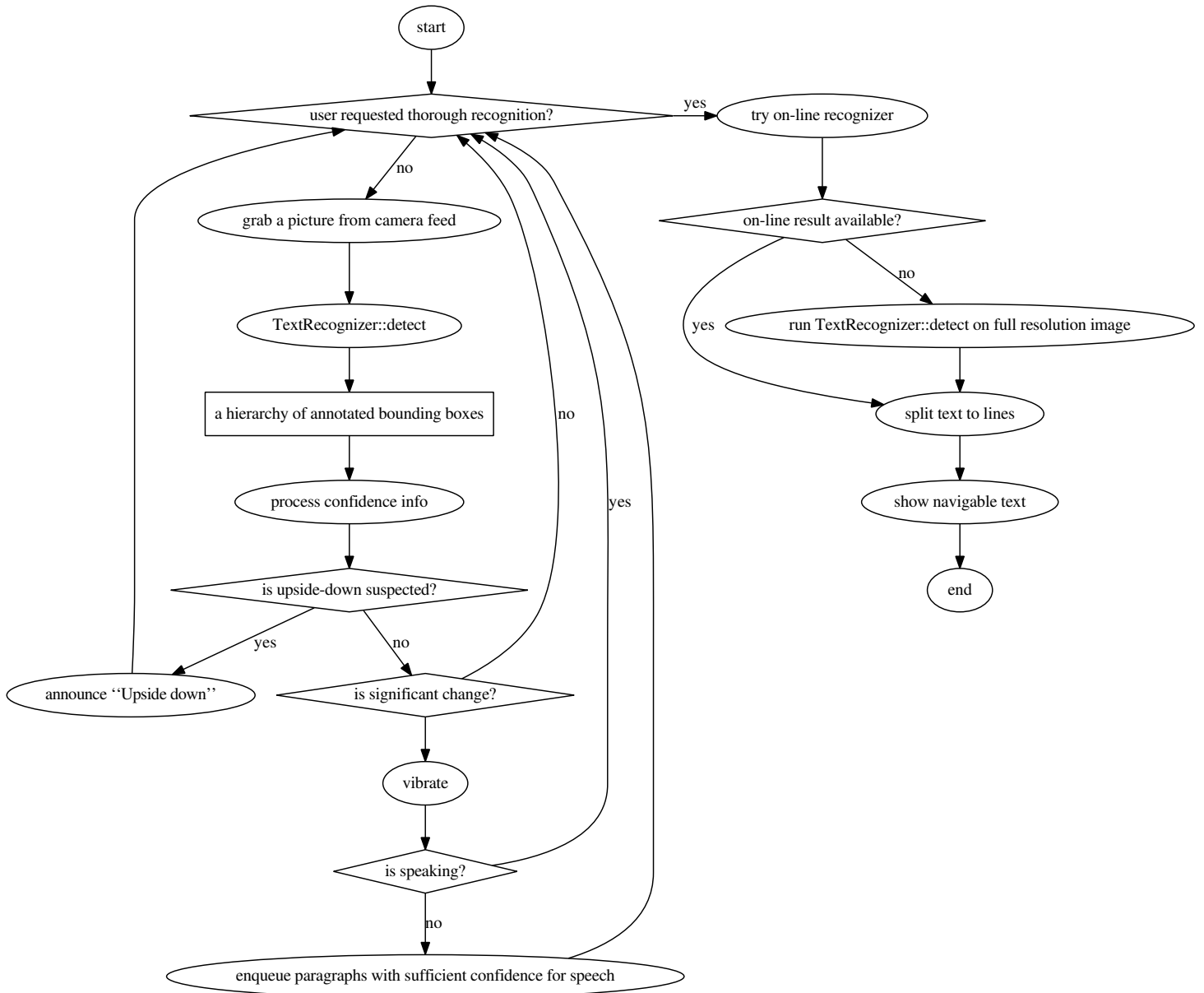
The recognition results from Microsoft Cognitive or TextRecognizer are then linearized and split to paragraphs. Each paragraph is presented on a separate screen. User can navigate between screens using standard gestures of the *Core System*.

## 5.3. Conclusion

A simple OCR application using off-the-shelf OCR engines was implemented. It is able to run in both on-line and off-line modes, improving the results by using a remote OCR service when Internet connection is available.

While this application has no ambitions to replace the standard flat-bed scanner + PC + ABBY Fine Reader [59] setup for reading books and longer documents, such mobile OCR solution can be successfully used in situations where a more immediate result is desirable, like identification of unknown objects where no alternative means of identification exists or are practical (delivered mail, receipts, bags with spices, cosmetic products, book covers etc.).

Some real-life recognition results are presented in section 5.4.



**Figure 17.** The workflow of proposed OCR application. The left subgraph depicts the “camera/document positioning mode” and the right one the “reading mode”

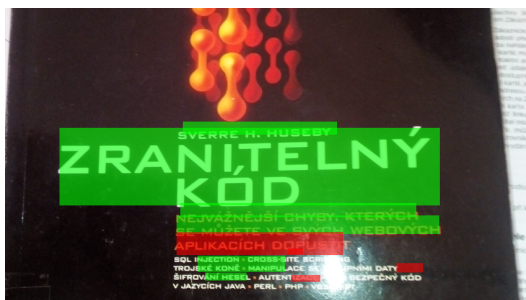
## 5.4. Examples of recognition results

This section demonstrates the real-life performance of the text recognition feature. The green bounding boxes signify high confidence regions. These regions are read aloud to user in real-time. Red bounding boxes depict low confidence regions that are ignored. Two recognition results are shown, first one is immediate real-time recognition (**I**), the second one is final recognition result (**F**).



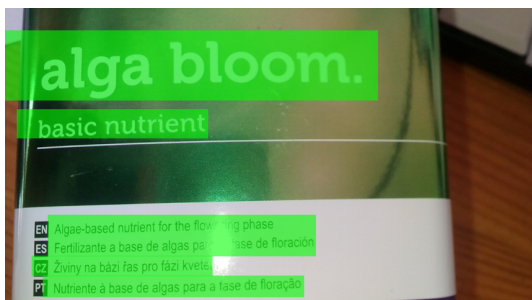
**I:** IAESTE Czech Republic Let us light your fire

**F:** IAESTE Czech Republic Let us light your fire



**I:** ERRE H. HUSEBY ZRANITELNÝ KOD KTERÝCH SE MUŽETE PICH JAVA n PERL PHP VBSCRIPT

**F:** ZRANITELNÝ KOD SVERREH. HUSEBY NEJVÁŽNĚJŠÍ CHYBY' ÁRÝCH SE MŮŽETE VEI APLIKACÍCH DOPU SQL INJECTION CROSS-SITE scRIPTINq TROJSKÉ KONĚ MANIPULACE SE VSTU DATY ŠIFROVÁNÍ HESEL AUTENTIZCE ĚRO-BEZPEČNÝ KÓD V JAZYCÍCH JAVA PERL PHP VBSCRIPT



**I:** alga bloom. tbasicnutrient EN Algae-based nutrient for the fowering phase E5 Fertilizante a base de algas para la fase de floración UVI na bázi fas pro fázi kvetení Nutriente à base de algas para a fase de floração

**F:** alga bloom. basic nutrient EN Algae-based nutrient for the flowering bhese ES Fertilizante a base de algas para la fase de floración C2 Živiny na bázi řas pro fázi kvetení Nutriente à base de algas para a fase de floração



**I:** vyrobeno z českého mléka

**F:** Kefírové mléko vyrobeno z českého mféka



## 6. Camera and gallery application

### 6.1. Motivation

A part of the task of this thesis was to develop an accessible solution for camera and gallery. While the camera and image gallery function may seem to be inaccessible “by definition”, there are important reasons why a person with severe vision loss might want to use it:

- people with no useful vision may want to record and significant moments or interesting places for their sighted relatives and friends
- individuals with low-vision conditions may want to use the camera as a magnifier or apply some high contrast filters to help them distinguish features and objects in images more easily
- some visually impaired individuals may find it useful to take a picture of an unknown object and share it with a sighted friend or use a machine learning recognizer to recognize the image

### 6.2. Design of the camera application with assistive features

The camera application is based on the new Camera 2 API (`android.hardware.camera2`) introduced in Android 5.1. The base for the application was the official example given by Google [60]. This example was then modified to accommodate assistive features described in this section.

The basic workflow of the camera can be summarized in three steps:

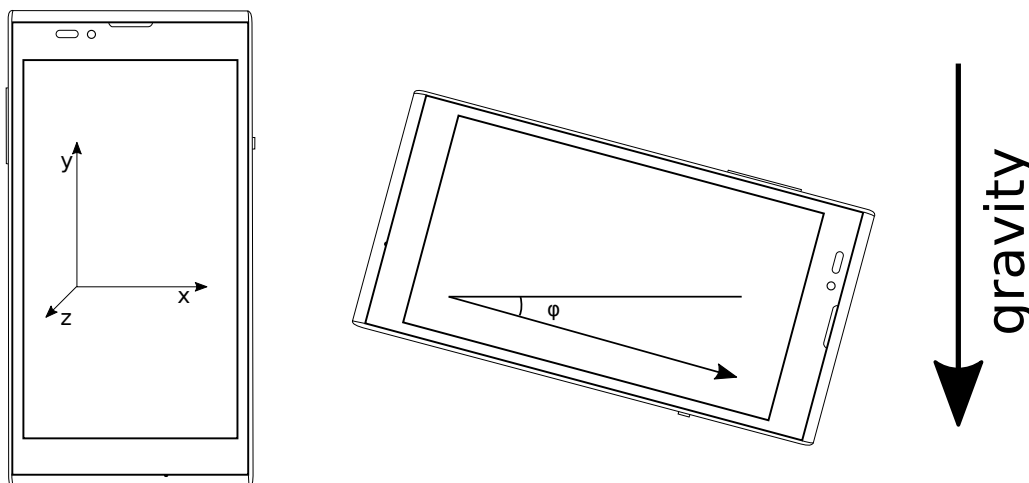
1. live preview (with optional audible feedback)
2. capture of the image itself
3. labeling and storage of the image

A more detailed work-flow graph is depicted in Figure 21.

### 6.3. Assistive features for image capture

#### 6.3.1. Face detection

Since the first version, Android comes with off-the-shelf face detection algorithm in class `FaceDetector`. This algorithm is able to detect multiple faces in real-time camera feed. The face detection algorithm is not the classic Haar cascade, it is an algorithm developed by Neven vision, later acquired by Google[61]. The implementation is open-source and available in the Android source code tree, but is patented[62] and not very well documented.



**Figure 18.** Accelerometer axes and the  $\phi$  angle.

Since face detection may be a useful feature for portraits and self-portraits, it was included in the program.

### 6.3.2. Tilt detection

Tilt detection with auditory feedback is a crucial feature for blind photography. This task can be efficiently done using an accelerometer, which is embedded within the *Core System*.

Android API provides native support for accelerometers. There are three axes where acceleration is measured, as depicted in Figure 18:

- **x** axis points in the direction of screen width
- **y** axis points is in the direction of screen height
- **z** axis which is in the direction of phone “depth” or “thickness”

Since the *Core System* only has a 2-axis accelerometer, only **x** and **y** axes could be used. The **z** axis is ignored and is assumed that it is tangential to the earth surface. Under those assumptions, we can compute the tilt angle in radians as  $\phi = \text{atan2}(d_x, d_y) + \pi$ .

The computed  $\phi$  angle is then used to distinguish between four basic orientations:

1.  $\phi \in \left[ \frac{\pi}{4}, \frac{3\pi}{4} \right] \rightarrow$  landscape
2.  $\phi \in \left[ \frac{3\pi}{4}, \frac{5\pi}{4} \right] \rightarrow$  portrait
3.  $\phi \in \left[ \frac{5\pi}{4}, \frac{7\pi}{4} \right] \rightarrow$  landscape, upside down
4. otherwise  $\rightarrow$  portrait, upside down

The coarse orientation is announced by text-to-speech every time it changes.

After one of those four coarse orientations is established, relative angle deviation from the basic (landscape, portrait, ...) orientation is needed for more precise acoustic tilt feedback. This relative value is obtained by subtracting  $\frac{\pi}{2}, \pi, \frac{3\pi}{2}$  and  $2\pi$  respectively, depending on the coarse orientation detected above.

The fine tilt is announced using a tone interval. The program contains a look-up table  $T$  containing tone frequencies from C4 (262 Hz) to B8 (7902 Hz). C5 (523 Hz) was selected as the reference tone  $t_0$ . The interval played to the user always consists of the base tone and some tone from the look-up table. The frequency of the second tone  $t_1$  is computed as follows:

$$t_1 = T(\min(\lfloor 50 \cdot \tan(|\phi|) + \text{idx}_T(t_0) \rfloor, T_{max}))$$

where  $\text{idx}_T(t_0)$  is the index of  $t_0$  in  $T$  and  $T_{max}$  is the last tone in  $T$ .

The tones are then played in order  $t_1 \rightarrow t_0$  for  $\phi < 0$  and in order  $t_0 \rightarrow t_1$  otherwise, giving a falling or rising musical interval respectively. The tones are played continuously in a loop, once every 1.5 seconds, giving an immediate feedback about camera tilt.

## 6.4. Labeling and saving of the image

Once the user presses the volume rocker, the capture sequence is started. The beginning and end of the capture sequence is announced by distinct sounds, so that the user knows when to hold the phone steady. The user is then presented with a classical the *Core System* menu where they can select one of the following options:

1. Save image
2. Recognize image
3. View image

### 6.4.1. Blur detection

After an image is taken, blur detection is performed to warn the person taking the image the picture may be too blurred. There is an easy yet effective heuristic to detect blurred images, which is based on variance of Laplacian.

First published in [63] and popularized by the website PyImageSearch[64], the method convolves the greyscale image with a Laplacian kernel and then computes the variance statistic on the resulting array. The single float number resulting from this computation can then be used to decide if the image is blurry or not, by comparing it to a threshold value. From experiments with the phone camera, the value of 100 seemed to work reasonably well. I.e. if the variance of Laplacian is less than 50, the user is warned about the picture being blurry before the menu is displayed, and they can choose to take the image another time. Example of blur detection can be seen on Figure 19.

### 6.4.2. Support for image recognition

Experiments with various image recognition techniques were performed. In the end, the most successful candidates for use in the *Core System* were Microsoft Cognitive image recognition and CloudSight, which is the service behind TapTapSee mentioned in Chapter 2.



**Figure 19.** An example of blur detection. [64]

### Microsoft Cognitive API

Microsoft Cognitive Services [53], already discussed in Chapter 5 offers an on-line service for image recognition, under similar conditions to their OCR service (5000 requests per month for free). Unlike for OCR, Microsoft published concrete papers and algorithms behind this service [65] [66]. The algorithm used for this service won the 2015 CVPR COCO Image Captioning Challenge. However, the subjective results of this are more general than results provided by the CloudSight service and not so useful for use in an assistive tool.

The functionality is accessible via a REST API similar to the one used for OCR. The Snippet 3 shows an example of a raw returned result. The results contain a caption, which is a simple sentence describing the image, and a list of tags.

**Snippet 3.** A sample image description result returned by Microsoft Cognitive

```
{  "Tags": [
    "table", "food", "indoor", "sitting", "top", "doughnut",
    "small", "plate", "donut", "wooden", "white", "sandwich"
  ],
  "Captions": [{
    "Text": "a close up of food on a table",
    "Confidence": 0.8934385
  }]
}
```

Only the description is currently displayed to the user. Recognition times are usually around 10 seconds and only a single HTTP request is needed (unlike CloudSight).

### CloudSight API

CloudSight is a company primarily focusing on image classification and captioning. Their flagships are TapTapSee and CamFind applications, but the technology is also

available in a form of API for any third-party application [67]. The prices are rather high, there is a free trial account available, which offers 500 image requests and requires a credit card verification.

The main advantage of CloudSight is that it gives descriptions more suitable to visually impaired people, including product names, text labels content etc. The disadvantage compared to Microsoft Cognitive is that it has much longer processing time, is more expensive and their servers are sometimes overloaded (only trial version tested), so no caption will be produced at all.

The communication with this API is a bit more complicated. Due to longer processing times (usually between 20 and 30 seconds), the request is split two HTTP calls:

1. The image is uploaded via a multi-part POST request. The POST request contains the image in JPEG and optionally some additional metadata, like GPS coordinates and coordinates of the focused region in the image. A response to this POST request looks like in Snippet 4 and contains a token which is later used to retrieve the result.
2. The program then periodically issues a GET request containing the token. The server will reply by a JSON structure containing either "status": "completed", "status": "not completed" or "status": "timeout" fields. The client application should actively wait for the completed status and then present the result. An example of the final recognition result is in the Snippet 5.

**Snippet 4.** A response from CloudSight after issuing an image recognition request.[67]

```
{
  "url": "//images.cloudsightapi.com/uploads/image_request/image
/19/19404/19404152/Image.jpg",
  "token": "AJKAWHKGLjqMd9KDNIXQfg",
}
```

**Snippet 5.** A final response from CloudSight, containing image description.[67]

```
{
  "status" : "completed",
  "name" : "red beats by dre headphones",
  "flags" : ["adult"],
  "ttl" : 60,
  "url" : "//images.cloudsightapi.com/uploads/image_request/image
/19/19404/19404152/Image.jpg",
  "token" : "WySLTJWESPTtt6v0oBmzKf"
}
```

Both APIs were implemented and the service to use can be selected by switching a constant at application compile time. The final application will most likely use Microsoft API, because it is more stable, cheaper and the recognition does not take that long.

### 6.4.3. Voice captioning

A substantial function of blind photography is image captioning. After few experiments, the following approach was chosen: Users are prompted to record (up to) a 5 second voice caption after they choose the "Save image" option from the menu. This recording is later presented in the image viewer.

### Recording

Recording is achieved using standard Android means (`AudioRecord`). To achieve a clear, understandable and normalized sound, `Android NoiseSuppressor` and `AutomaticGainControl` sound filters are used during recording of the voice caption.

### Encoding and storage

This voice note is then encoded using Opus codec [68] and saved directly in the Exif metadata of the image using the Exif library [69]. While the Exif standard contains a specification of embedding sound [70], it is not widely supported and uses obsolete codecs with poor results, that's why a custom solution was chosen for caption storage.

The Exif format supports custom metadata tags, so a new tag with an ID value 9990, which is well above the commonly used tags was added to the Exif library. The data is stored in pure binary format, as produced by the Opus codec.

### Other metadata

Some other important metadata is also added during the saving process:

- date and time
- GPS coordinates (if available)
- phone orientation (landscape, portrait)
- results of a recognizer (if used during the saving process)
- information about camera (manufacturer, focal length, etc.)

## 6.5. Design of an image gallery application with assistive features

No mobile camera implementation would be complete without a way to actually browse the images and work with them. The specifics in this case are that the application must be usable even to users with no useful sight. A prerequisite for this is good quality captioning, which is achieved using the voice captioning function presented in the previous section.

The work-flow graph of implemented image gallery can be seen in Figure 22.

### 6.5.1. Browsing all images

The easiest way to browse images is to browse all. The images are presented in reverse-chronological order (newest first). Images are presented to user the same way as any other menu of the *Core System*

### 6.5.2. Browsing by category

In the image gallery settings, users can define arbitrary categories and sort images into them. Categories work as “labels”, so an image can be assigned to more categories or none at all. In the “browse by category” menu, user is presented with a list of categories and they can pick one of them. A list consisting of all images with a given category is then presented.

The list of all categories is stored in a Java serialization format in the private application directory. Each category record consists of a randomly generated UUID and a category name. Categorized images then contain a new Exif tag with ID value 9991, which contains a JSON[55] array of category UUIDs. This way, users can rename categories arbitrarily and the change is automatically reflected in all images.

### 6.5.3. Browsing by date

The last mode is browsing by date. The image Exif date is used for this purpose. User is presented with the following menu, where they can list only images taken in a particular time. Only those time categories that contain any images are presented to the user.

- today
- this week
- this month
- 2017
  - May
  - April
  - February
  - ...
- 2016
  - December
  - ...
- ...

The year menus are two-level and allow to filter pictures with month granularity, the rest is a single-level.

### 6.5.4. Low-vision filters

Inspired by [24], three “low-vision filters” have been implemented for usage in the image viewer.

- global thresholding based on Otsu method
- edge enhancement using Sobel edge detector
- high contrast

Examples of the output of these filters can be seen on Figure 6.5.4. The filters implemented are loosely inspired by [25]. However, the usability testing and later consultation with some low-vision people performed after implementation of the filters have shown that visually impaired people rarely find these filters useful for images and prefer unfiltered images. One of these filters can be optionally activated in settings.

6. Camera and gallery application



**Figure 20.** Implemented low-vision filters: original image, Otsu threshold, Sobel edges, high contrast



6.5. Design of an image gallery application with assistive features

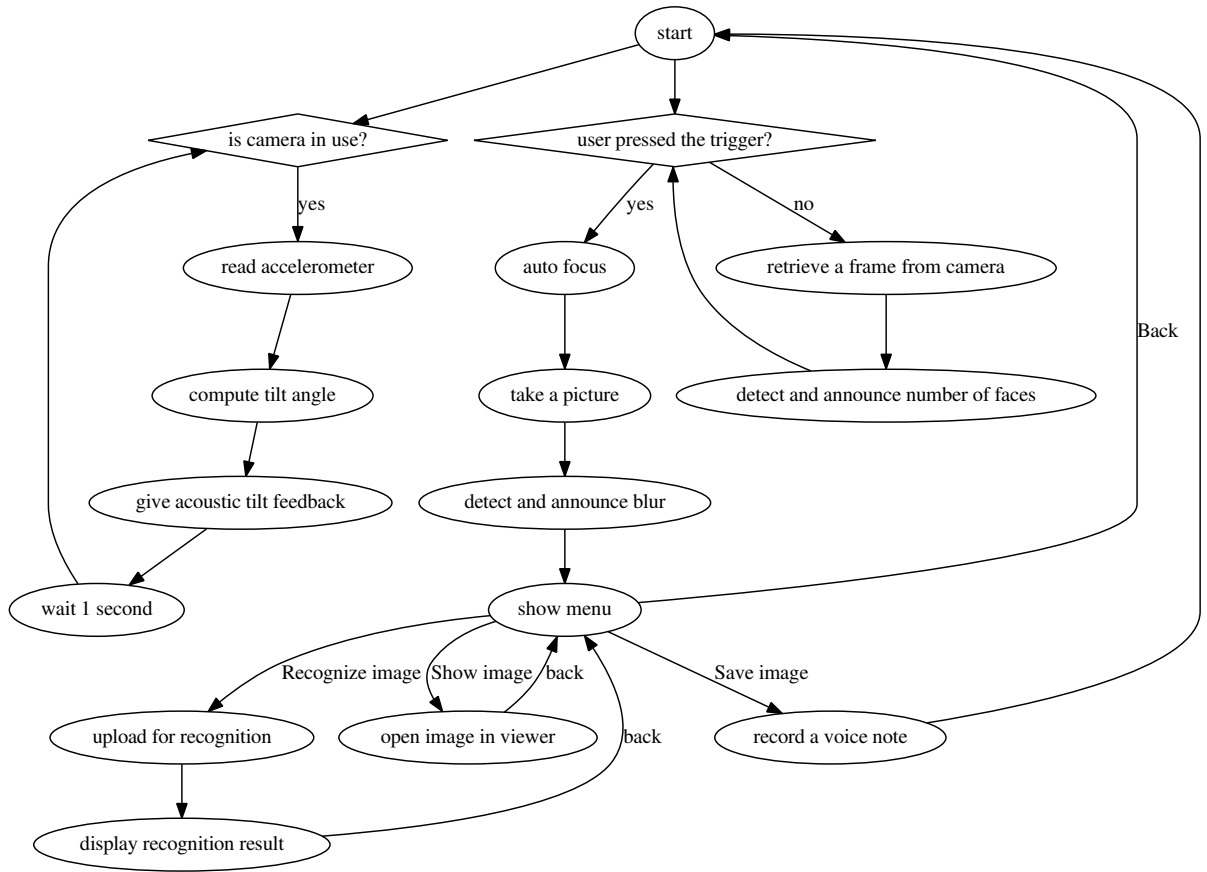


Figure 21. The workflow of proposed camera application.

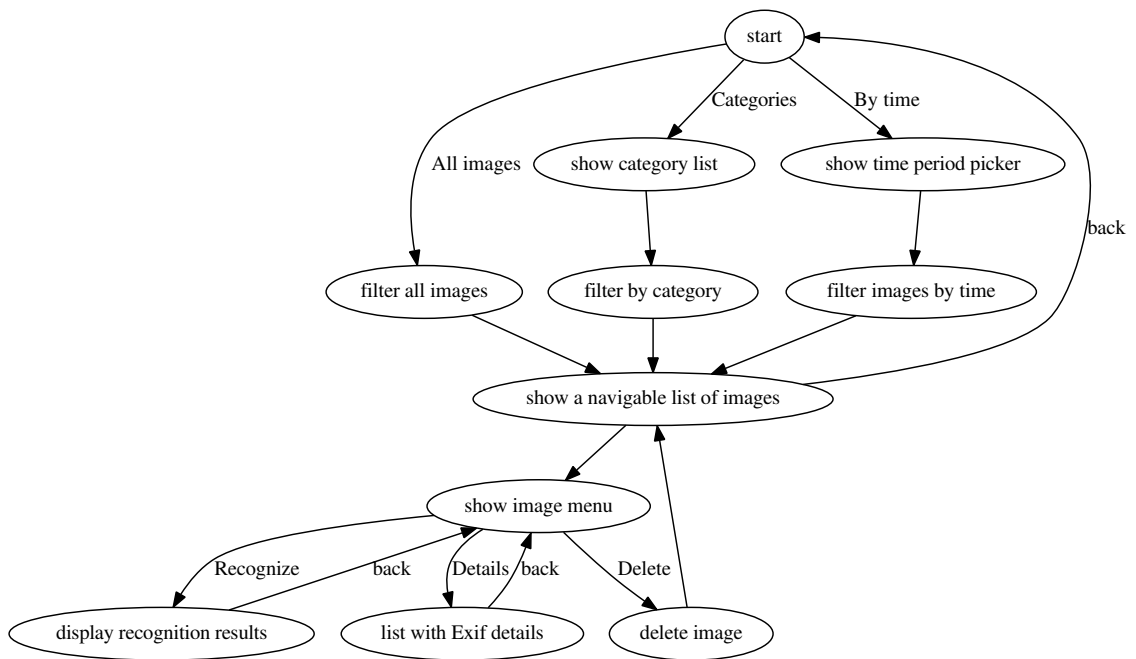


Figure 22. The workflow of proposed camera application.

## 7. Testing

This chapter documents usability testing of suggested accessibility tools. Usability testing is a technique used in user-centered interaction design to evaluate a product by testing it on users.[71] The purpose of this testing is to discover usability problems or sub-optimalities in a product (in this case a computer program) by testing it on volunteers of the target group. The participants are encouraged to “think aloud” and to give a sincere feedback.

The solution presented in this work is targeted towards users with vision loss level ranging from intermediate vision impairment to no light perception. It is assumed, that users with a better visual function would unlikely require a special solution for a mobile phone.

The participants of this experiment were contacted with the help of Czech Blind United (SONS). There were 6 participants in total, three men, and three women. All of them were familiar with the *Core System* already, two of them were the *Core System* beta testing users (though not familiar with the tested applications at the time of testing). The responses to pre-test and post-test questionnaires are enclosed in Appendix A

The testing process was performed in three phases:

1. Pre-test questionnaire is administered right before the test. The goal of this questionnaire is to collect important details about the participants and to get to know their expectations and interests relevant.
2. The testing itself, which consists of a set of tasks that have to be completed sequentially. There is a test moderator who reads aloud the tasks and follows the progress of the participant.
3. Post-test questionnaire is administered immediately after completion of the testing, to collect feedback and immediate ideas of the participant.



**Figure 23.** Blind photography: These two pictures were taken during the testing by participants who had no light perception, guided only by assistive features of the camera application.

## 7.1. Pre-test

1. Your age?
  - a) less than 20
  - b) 20-35
  - c) 35-50
  - d) 50-65
  - e) more than 65
2. Your gender?
  - a) male
  - b) female
3. Do you have a visual experience? (=Have you ever seen?)
  - a) yes
  - b) no
4. How would you rate the level of your visual impairment from the functional perspective?
  - a) intermediate vision impairment (able to read with reading glasses or optical magnifiers, navigates mostly visually)
  - b) serious visual impairment (have some limited projection, able read only with difficulties and with the help of digital magnifiers, navigates with the help of a white cane)
  - c) blindness with no useful projection, unable to read at all, able to distinguish light from darkness
  - d) no light perception at all
5. What accessibility solution you use for your mobile phone?
6. Do you use mobile data on your current mobile phone? Would you consider getting a data plan if it was required for some accessibility features, like text or image recognition?
  - a) yes
  - b) no, but would consider it
  - c) no, and would not consider it
7. Do you like the idea of integrating visual accessibility tools into a mobile phone?
  - a) yes
  - b) no
8. Would you find it helpful take pictures with your phone and share them with your sighted relatives or friends?
  - a) yes
  - b) no
9. Which of the following accessibility features you'd likely use if they were integrated into your phone?
  - a) color recognition
  - b) banknote recognition
  - c) text recognition (OCR)
  - d) digital video magnifier
  - e) digital image enhancement for captured still images (high contrast, inverse, color filters, edge enhancement)
  - f) general object recognition
  - g) object labeling with help of optical markers (bar codes)
  - h) illumination level detection
  - i) LCD display recognition

## 7. Testing

- j) recognition of clothes
- k) possibility of remote live audio/video conference with a sighted person to help you with some task (relative, assistant, friend)

## 7.2. Tasks

Since users with a variable level of vision loss were participating in the experiment, tasks had to be prepared so that they were relevant and possible to finish to all of them. The tasks were split into three sub-sections: text recognition, camera and image gallery, which were performed more or less independently.

### 7.2.1. Text recognition

1. Find the accessibility tool called “Text recognition” in the phone menu.
2. Slide one finger from the top of the screen to bottom to get a short context help for this function.
3. There are three closed bags with spices in front of you. Use the application to find a bag with “Oregano”.
4. The other side of the bag with “Oregano” contains a recipe for “Italian Mushrooms”. Try to locate the text in the view and use the “thorough recognition” function to get the text of the recipe. Skim through the obtained text.
5. long press the lock button to get to the home screen

### 7.2.2. Camera

1. Find the function “Camera” in the menu and read the help
2. Run the camera and try out the tilt feedback function.
3. Run the camera and try to take a picture of a person. The voice feedback will notify you when there is a face in the image.
4. Try out the function “image recognition”.
5. Save the resulting image, include some meaningful voice caption
6. Return back to the “Media” menu

### 7.2.3. Pictures

1. Find the function “Image Gallery”.
2. Try to find the picture you have taken among the others.
3. From the context menu, figure out the size of the image in pixels.
4. Delete the image.

### 7.2.4. Banknotes

1. Open the function “Banknote recognition”
2. Read the help of the application to get familiar with its control.
3. There is a banknote in front of you. Use the application to recognize its value.

## 7.3. Post-test

1. Did you find the tasks difficult?
2. What was the hardest part?
3. How you like the “Text Recognition” application? Any feedback or ideas for improvement?
4. How you like the “Camera” application? Any feedback or ideas for improvement?
5. How you like the “Gallery” application? Any feedback or ideas for improvement?

## 7.4. Important findings of the testing

### 7.4.1. Text recognition

- two users requested the possibility to **save recognized text** in Notes
- all users except one were satisfied with text **recognition accuracy**
- most users find it faster to use such mobile solution (rather than using a PC+scanner) for the **purpose of identifying objects**
- all users **were able to use the OCR function to pick the right bag of spices** from a set of three bags, randomly shuffled and rotated
- half of the users **succeeded in reading the recipe** on the back of the bag, all of them would rather use a flat-bed scanner for this task
- all users except one welcomed the **dual-mode function** (immediate reading vs. thorough analysis)
- some participants **struggled with correct positioning** of the camera in immediate reading mode (mostly with distance) and needed further instructions

### 7.4.2. Camera and images

- while **blind participants liked the voice labeling** feature in the camera and gallery, **low-vision users would prefer the labels to be optional or in text only**
- two users asked for a **change of tilt detection feedback to vibrations** (possibly optionally)
- all users, including the blind ones **succeeded in taking a picture of the test moderator** using assistive features in the camera application
- all users would welcome a function to **share images by e-mail or MMS**
- the low-vision users would like to have digital zoom in the camera capture activity

### 7.4.3. Banknotes

- all participants **were able to recognize a 100 CZK note successfully**
- four participants spontaneously mentioned **not needing a tool for banknote recognition**
- two users mentioned usefulness of such application when traveling abroad, if **foreign currencies** were supported
- two participants complained that the recognition of the banknote **taking too long**

### 7.4.4. General findings

- four of six users liked the idea of **tagging daily objects using QR codes** and in-phone database
- low-vision users were **not seriously interested in “image enhancement”** of any kind
- majority of users **like the idea of integrating accessibility tools** into a mobile phone, one subject thinks dedicated tools are usually better

## 7.5. Testing summary

Six volunteers with visual handicaps ranging from intermediate low-vision to no light perception participated in the testing. All of them were already familiar with basic control of the *Core System*, and they found the user interface of new tested applications intuitive. The feedback on new applications was mostly positive. Participants quickly understood the basic control and function of various features. The worst stumbling block was understanding the correct way to hold the phone during the OCR task, where three of six participants initially struggled, but after more thorough instructions were all of them able to use the OCR function for text recognition successfully.

All participants were positive about proposed applications and solutions except for few details mentioned in the previous section. These suggestions will be reflected in future development. The banknote recognition function turned out to be not very useful in principle, because most visually impaired people in Czech Republic, where banknotes can be easily distinguished by their size, are able to do so. Participants would welcome if the banknote recognition was faster, which should be possible to achieve if the algorithm was parallelized.

## 8. Technical details

This chapter describes some technical details of the solutions described in previous chapters. This chapter refers to files and folders located on the enclosed CD. A complete description of the CD content can be found in Appendix B.

The implementation consists of three mutually independent modules: camera+gallery, text recognition and banknote recognition.

### 8.1. Technologies behind the *Core System*

The *Core System* consists of:

1. the main module, which is a standard Android 5.1 (API 22), which is written in Java 7
2. a module for Xposed Framework (written also in Java 7), which enables the programmers to dynamically modify some aspects of the target operating system
3. some third-party native libraries
4. some helper applications (on-line update helper, text-to-speech) and customizations in the phone ROM

### 8.2. Modules implemented as a part of this thesis

#### 8.2.1. The Banknote recognition engine

Banknote recognition module consists of two parts. The first part is used for dataset augmentation and training of the classifier and is implemented in Python 2.7 for versatility and ease of modification and prototyping. The resulting models are stored in OpenCV `FileStorage` XML format and compressed by GZIP for compactness (albeit not documented, OpenCV has direct support for handling GZIP-compressed storages). Source codes of this training can be find in `python` folder.

The banknote recognizer running directly on the *Core System* is implemented in native C++ 11 with the help of the OpenCV library and is connected with the rest of the application using the standard JNI way. The user interface is implemented in Java 7 and integrated with the rest of the *Core System*. Source codes of the C++ and Java part are in `cxx/banknotes` and `java/coresystem/aids/banknotes` directories respectively.

### 8.3. Text Recognition

The Text Recognition module is implemented in Java and uses the native `TextRecognizer` library provided by Google Play API. The sources can be found in `java/coresystem/aids/ocr`.

### 8.4. Camera

The `Camera` and `gallery` modules are also implemented in Java 7 using `Camera2` API of Android. Low-vision filters as well as blur detection is implemented in native C++ with OpenCV with JNI Java wrappers. The sources can be found in `java/coresystem/camera` and `java/coresystem/images`, the implementation of the filters can be found in `cxx/coresystem/common/opencv-android.cpp`.

### 8.5. A list of third-party libraries used

- OpenCV 2.4 library for image processing and pattern recognition algorithms, native C++ and Python bindings are used [16]
- Google Play Services (for text recognition) [72]
- `com.android.volley` – for all HTTP communication [56]
- `top.oply.opuslib` – for encoding of sound comments [73]
- `it.sephiroth.android.exif` – an extensible library for Exif manipulation [69]
- CloudSight Java client – a library to simplify work with the CloudSight API [74]
- Numpy – for matrix computations in Python [75]
- Scipy – for statistical functions in Python [76]
- Matplotlib – for plotting during classifier development [77]
- Levenshtein – for approximate median string computation [41]



## 9. Conclusion

*When I lost my sight, Werner, people said I was brave. When my father left, people said I was brave. But it is not bravery; I have no choice. I wake up and live my life. Don't you do the same?* – Antony Doerr, *All the Light We Cannot See*

### Visual impairment in the human history

Blindness as a handicap has been naturally occurring in every society in the history. During the most of the history, visually impaired people (and all disabled people for that matter), have been considered to be a social burden and if not taken care of by their families, most would perish in the harsh environment. Blind babies would be abandoned, and blind adults would often end up as beggars.

By the Enlightenment, societies began to acquire a belief that there was an obligation to help the less fortunate, including the blind. It seems now obvious, that the most valuable support, that can be offered to those less fortunate, is to teach them how to compensate for their disabilities.

A (usually white) cane, the simplest yet unsurpassed assistive tool for the blind, has been the symbol of blindness for centuries, later accompanied by the tactile writing system developed by Louis Braille, which greatly supported the spread of literacy in the blind community. The ability to read and write, which the blind community regained with Braille's invention, was the first step to education, emancipation, and self-sufficiency of blind individuals.

### Blindness and technology

Attempts to fix blindness or at least mitigate its impact on a blind person and the people around them have certainly always been there. While many diseases that would lead to practical blindness in the past are easily curable with modern medicine (e.g. cataracts), a definitive cure for blindness, like an eye transplant, stem cell therapy or a visual prosthesis comparable to (even a very bad) human eye, are still beyond our grasp. Nevertheless, even for incurable vision problems, we can still develop engineering solutions, modern “white canes” of a sort – digital accessibility tools.

Computer vision, as a scientific field, is experiencing unprecedented development, mostly due to advances in deep learning. Many things that have long been considered impossible, like automatic image captioning, are now a reality. We also live in the golden age of mobile technology, and the computational power of devices we carry in our pockets

## 9. Conclusion

is growing with every generation, surpassing previous generations of desktop computers in their computational and storage capabilities.

A naive assumption might be, that given all this, mobile accessibility tools for the visually impaired should be an ideal flagship for computer vision algorithms. In reality, the interdisciplinary cooperation between developers of accessibility tools and computer vision research is less than ideal.

Many technologies and algorithms that would be applicable in the field of assistive technology are not being used there yet in practice. The choice of accessibility tools that use the state-of-the-art technology is limited. The main reason seems to be that development of computer vision algorithms is primarily motivated by academic (in the case of university research teams) and economic (in the case of private companies) goals, not so much by real needs of people. Also, awareness of ordinary people about blind and partially sighted people is inadequate, considering how common blindness and low-vision conditions are.

## Contribution of this work

This work is a humble attempt to contribute to bridging the gap between science and assistive technologies. The author of this thesis believes, that visually impaired people can live independent lives, and given the right tools and training, can work and can be a productive part of the society.

This manuscript has introduced three innovative accessibility tools:

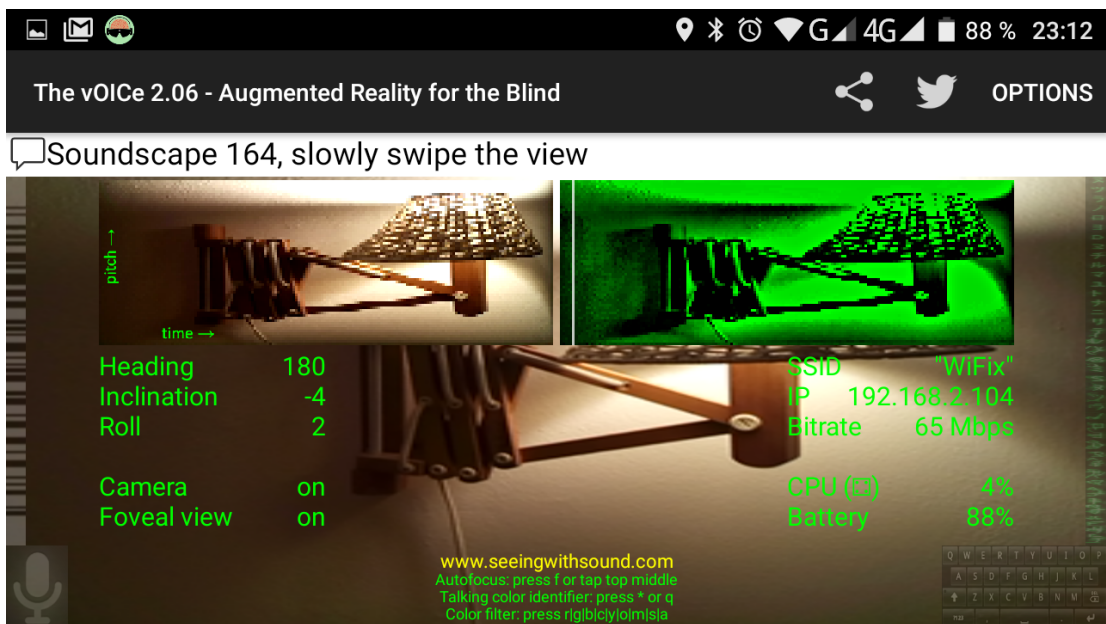
1. a banknote recognition engine and application based on BRISK detector/descriptor and Gradient Boosted Trees
2. a simple photo-OCR application harnessing two state-of-the-art text recognition engines and empowering the blind to read labels on household objects and identify them
3. a camera application with assistive features, support for two different cloud image recognition services, voice captioning of the photos and a simple yet powerful image browser featuring simple categorization and low-vision filters

It is important to note that all tools were developed in close cooperation with experts from the Czech Blind United (SONS) and went through usability testing. The results are perceived positively by the target group.

## Future work

One of the concepts that could be covered in this thesis and were not, is the principle of sonification, as introduced by the vOICe project [78] [79]. This project currently offers a free Android application (see Figure 24) and a simple sonification implementation in open-source C++.

The vOICe project focuses on converting images into so-called “soundscapes”, which in the most basic principle consists of taking a greyscale image as if it were a sound spectrogram and computing inverse short-time Fourier transform to convert a picture into a binaural (3D) sound. Paired with special glasses and headphones, this concept



**Figure 24.** The user interface of vOICe for Android.

could turn a mobile phone into a visual prosthesis.

Authors of this concept claim that with enough training, their testing subjects were able to achieve “artificial synesthesia”, i.e. they effectively restored the sense of “seeing” (in a very low resolution) with the help of this algorithm. Such algorithm would be a very interesting way of viewing the images on a mobile phone for blind people, though the training required to make sense of the sounds might be prohibitive.

Other, more mundane ideas for future work would be:

1. improvements in banknote recognition speed
2. more research into the low-vision filters
3. better upside-down detection for OCR
4. implementation of a segment display recognizer
5. implementation of QR code object labeling

Most of which were already discussed more in detail in the Chapter 7.

# Appendix A.

## Testing results

### A.1. Pre-test questionnaire

	M	P	H	V	S	R
1 age	c	b	c	d	d	c
2 sex	male	female	male	male	female	female
3 tester	yes	yes	yes	yes	yes	yes
4 visual xp	yes	yes	yes	yes	yes	yes
5 sight loss	d	a	c	d	b	c
6 phone type	A+Tb, I+Vo	A+Mag	I+Vo	Cs	I+Vo	Cs
7 assist	a	c	a	a	a	a
8 without net	a	b	b	b	b	a
9 share images	yes	yes	yes	yes	yes	yes
10 features	a+c+f+g +h+i+j+k	c+d+e+k	c+f+g +h+k	a+c+f+g +h+i+j+k	a+c+d +f+i+k	a+b+c+g +h+i+j+k

Legend:

- P3: m = male, f = female
- P5
  - A+Tb = Android with TalkBack service
  - A+Mag = Android with Magnifier
  - I+Vo = iPhone with VoiceOver
  - Cs = the *Core System*

### A.2. Post-test questionnaire

#### A.2.1. Subject M

1. “All tasks were extremely easy.”
2. “Horizon feedback in camera, it is unintuitive.”
3. “Works well, within realistic expectations. Upside-down detection gives too much false positives.”
4. “Camera horizon feedback makes annoying beeps and is unintuitive. It is a good function, but consider using vibrations instead. Otherwise works well. Likes the voice labeling feature.”
5. Works well, especially the voice notes are very good. It would be great if text labeling could be added too, possibly with automatic recognition of the voice labels.

6. "Works well, better usage instructions would be good though."

### A.2.2. Subject P

1. "All tasks were very easy, except for reading the recipe."
2. "OCR engine did not meet my expectations, it would not be helpful. The continuous reading function is not needed."
3. "Improve detection accuracy, at this point, it's not very useful for low-vision people."
4. "Camera horizon feedback is not needed and is unintuitive. Voice labeling makes no sense, text labeling would be preferred. Low-vision filters are not very useful."
5. "Basic, but works reasonably well. Improve the magnification function to make it more smooth."
6. "Not really needed for me, but seems to work."

### A.2.3. Subject H

1. "All tasks were easy."
2. "OCR accuracy is good, but needs better light conditions." (note: The subject H had serious problems with reflections from a light fixture.)
3. "It took some fighting to figure out the right orientation of the object and phone. Otherwise works well."
4. "Camera horizon feedback is too noisy, albeit useful. Consider a different method of horizon detection feedback. Voice labeling is very useful."
5. "It is simple, but works well."
6. "It is too slow. I can recognize banknotes easily without any tool."

### A.2.4. Subject V

1. "Tasks were easy. It takes some time to practice with a camera if you are blind though."
2. "Add the possibility to save recognized texts."
3. "The OCR is not suitable for longer texts, but that's expected. The accuracy is good enough for quick tasks like recognition of labels on daily products, very useful function."
4. "I like the sound feedback and face announcement. The application is easy to use."
5. "I like the possibility of recording my own notes and categorizing images. Sending by MMS or e-mail would be nice."
6. "I can't imagine using it in practice. I can recognize banknotes by their size. May be useful when traveling abroad though. The recognition time is a bit slow."

### A.2.5. Subject S

1. "It was easy. More practice would be needed though."
2. "I struggled with keeping the right distance from the text in the OCR task."
3. "The OCR application is good, I like the continuous reading function. Would not use it for longer texts, but it is very nice for short ones."
4. "The sound feedback seems useful and relatively intuitive, though a bit noisy. The face detection and blur detection is quite useful."

5. "The gallery works as it should. Sometimes is a bit sluggish though."
6. "Not really needed for me. But seems to work."

#### **A.2.6. Subject R**

1. "The tasks were easy to medium difficulty."
2. "I struggled with understanding the instructions how to hold the phone during the OCR task. Once I understood, it worked rather well."
3. "It is great, I like how it continually reads what's there. I believe I can get better at keeping the right distance from the object with practice"
4. "I like the sound feedback, it helps me keep the phone steady, though the tone is too aggressive sometimes."
5. "The voice notes labeling function is really useful for the gallery. I struggled with the categorization function initially."
6. "I've seen this function on the *Core System* before. It works okay, but support for more currencies would be handy."

## Appendix B.

### Content of the enclosed CD

- hadacja2-msc.pdf – This text.
- cxx – C++ banknote recognition engine.
- python – Python scripts used for banknote recognition algorithm training.
- java – Java source code of the *Core System* modules developed in this thesis.
- tex – L<sup>A</sup>T<sub>E</sub>X sources and images used to build this thesis.

# Bibliography

- [1] Pew Research Center. *Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies*. 2016. URL: [http://www.diapoimansi.gr/PDF/pew\\_research%201.pdf](http://www.diapoimansi.gr/PDF/pew_research%201.pdf).
- [2] World Health Organization. *Blindness*. 2017. URL: <http://www.who.int/topics/blindness/en/>.
- [3] World Health Organization. *Visual impairment and blindness*. 2016. URL: <http://www.who.int/mediacentre/factsheets/fs282/en/>.
- [4] World Health Organization. *Global Data on Visual Impairments 2010*. 2010. URL: <http://www.who.int/blindness/GLOBALDATAFINALforweb.pdf>.
- [5] Digital Trends. *Best Qwerty Phones*. 2016. URL: <http://www.digitaltrends.com/mobile/best-qwerty-phones/>.
- [6] Petr Svobodník. “Zpřístupnění mobilních telefonů se systémem Android pro nevidomé uživatele”. Diploma thesis. Czech Technical University in Prague, 2013.
- [7] Jan Pechan. “Detekce barev a bankovek pro nevidomé uživatele v prostředí Android”. Diploma thesis. 2015.
- [8] Google. *Google TalkBack Github repository*. 2016. URL: <https://github.com/google/talkback>.
- [9] Android Developers. *Accessibility*. 2017. URL: <https://developer.android.com/guide/topics/ui/accessibility/index.html>.
- [10] Android Developers. *Android API Reference: android.view.accessibility*. 2017. URL: <https://developer.android.com/reference/android/view/accessibility/package-summary.html>.
- [11] Kapsys. *SmartVision2 Product information*. 2017. URL: [http://www.kapsys.com/en/products/smartvision2/?noredirect=en\\_US](http://www.kapsys.com/en/products/smartvision2/?noredirect=en_US).
- [12] Claria Vision. *Claria Vox Product information*. 2016. URL: <https://www.claria-vision.com/en/>.
- [13] Stopka n.o. *COrvus, Accessible Kit for Android*. 2017. URL: <http://www.corvuskit.com/>.
- [14] *Volley library*. 2017. URL: <https://github.com/google/volley>.
- [15] *KNFB reader*. 2016. URL: <http://knfbreader.com/>.
- [16] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).
- [17] Ethan Rublee et al. “ORB: an efficient alternative to SIFT or SURF”. In: *In ICCV*.
- [18] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. 2nd ed. New York: Wiley, 2001.



- [19] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. *Efficient Graph-Based Image Segmentation*.
- [20] John Canny. “A computational approach to edge detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1986).
- [21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.
- [22] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. “GrabCut” – interactive foreground extraction using iterated graph cuts”. In: *ACM TRANS. GRAPH* (2004), pp. 309–314.
- [23] Gerald Friedland, Kristian Jantz, and Raul Rojas. “Siox: Simple interactive object extraction in still images”. In: *Multimedia, Seventh IEEE International Symposium on*. IEEE. 2005, 7–pp.
- [24] Howard Moshtael et al. “High tech aids low vision: a review of image processing for the visually impaired”. In: *Translational vision science & technology* 4.4 (2015), pp. 6–6.
- [25] Susan J. Leat et al. “Generic and customised digital image enhancement filters for the visually impaired”. In: *Vision Research* 45.15 (2005), pp. 1991–2007. ISSN: 0042-6989. DOI: <https://doi.org/10.1016/j.visres.2005.01.028>. URL: <http://www.sciencedirect.com/science/article/pii/S0042698905000787>.
- [26] Yangqing Jia et al. “Caffe: Convolutional Architecture for Fast Feature Embedding”. In: *arXiv preprint arXiv:1408.5093* (2014).
- [27] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: [10.1007/s11263-015-0816-y](https://doi.org/10.1007/s11263-015-0816-y).
- [28] Kelvin Xu et al. “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”. In: *arXiv preprint arXiv:1502.03044* (2015).
- [29] Jan Hadáček. “Detection and Recognition of Diacritical and Punctuation Marks in Real-World Images”. Bachelor thesis. Czech Technical University in Prague, 2014. URL: <https://cyber.felk.cvut.cz/theses/papers/444.pdf>.
- [30] L. Neumann and J. Matas. “Real-time scene text localization and recognition”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. 2012, pp. 3538–3545. DOI: [10.1109/CVPR.2012.6248097](https://doi.org/10.1109/CVPR.2012.6248097).
- [31] B. Epshtein, E. Ofek, and Y. Wexler. “Detecting text in natural scenes with stroke width transform”. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. 2010, pp. 2963–2970. DOI: [10.1109/CVPR.2010.5540041](https://doi.org/10.1109/CVPR.2010.5540041).
- [32] Xiangrong Chen and A.L. Yuille. “Detecting and reading text in natural scenes”. In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 2. 2004, DOI: [10.1109/CVPR.2004.1315187](https://doi.org/10.1109/CVPR.2004.1315187).
- [33] Dimosthenis Karatzas et al. “ICDAR 2015 competition on robust reading”. In: *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE. 2015, pp. 1156–1160.

- [34] Minghui Liao et al. “TextBoxes: A Fast Text Detector with a Single Deep Neural Network”. In: *arXiv preprint arXiv:1611.06779* (2016).
- [35] Hui Li and Chunhua Shen. “Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs”. In: *arXiv preprint arXiv:1601.05610* (2016).
- [36] Max Jaderberg et al. “Reading text in the wild with convolutional neural networks”. In: *International Journal of Computer Vision* 116.1 (2016), pp. 1–20.
- [37] S. Leutenegger, M. Chli, and R. Y. Siegwart. “BRISK: Binary Robust invariant scalable keypoints”. In: *2011 International Conference on Computer Vision*. Nov. 2011, pp. 2548–2555. DOI: 10.1109/ICCV.2011.6126542.
- [38] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “SURF: Speeded Up Robust Features”. In: *Computer Vision – ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I*. Ed. by Aleš Leonardis and Axel Bischof Horstand Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8. DOI: 10.1007/11744023\_32. URL: [http://dx.doi.org/10.1007/11744023\\_32](http://dx.doi.org/10.1007/11744023_32).
- [39] C. de la Higuera and F. Casacuberta. “Topology of strings: Median string is NP-complete”. In: *Theoretical Computer Science* 230.1 (2000), pp. 39–48. ISSN: 0304-3975. DOI: [http://dx.doi.org/10.1016/S0304-3975\(97\)00240-5](http://dx.doi.org/10.1016/S0304-3975(97)00240-5). URL: <http://www.sciencedirect.com/science/article/pii/S0304397597002405>.
- [40] Xiaoyi Jiang et al. “Dynamic computation of generalised median strings”. In: *Pattern Analysis & Applications* 6.3 (2003), pp. 185–193.
- [41] *Levenshtein library v. 0.12.0. Python extension for computing string edit distances and similarities*. 2014. URL: <https://pypi.python.org/pypi/python-Levenshtein/0.12.0>.
- [42] David Arthur and Sergei Vassilvitskii. “k-means++: The advantages of careful seeding”. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2007, pp. 1027–1035.
- [43] J Friedman. “Greedy Function Approximation: A Gradient Boosting Machine”. In: (1999).
- [44] Yoav Freund and Robert E. Schapire. *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*. 1997.
- [45] Tin Kam Ho. “Random decision forests”. In: *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. Vol. 1. IEEE. 1995, pp. 278–282.
- [46] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine learning* 20.3 (1995), pp. 273–297.
- [47] Ji Zhu et al. “Multi-class adaboost”. In: *Statistics and its Interface* 2.3 (2009), pp. 349–360.
- [48] Wikipedia. *Tf-idf — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-May-2017]. 2017. URL: <https://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&oldid=779803620>.
- [49] Martin A Fischler and Robert C Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

- [50] *Word Lens by Quest Visual*. 2017. URL: <https://questvisual.com/>.
- [51] Dan Bloomberg and contributors. *Leptonica*. 2016. URL: <http://www.leptonica.com/>.
- [52] *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from [tensorflow.org](http://tensorflow.org). 2015. URL: <http://tensorflow.org/>.
- [53] Microsoft Corporation. *Microsoft Cognitive Services*. 2016. URL: <https://www.microsoft.com/cognitive-services>.
- [54] Cesare Pautasso, Erik Wilde, and Rosa Alarcon. *REST: Advanced Research Topics and Practical Applications*. Springer Publishing Company, Incorporated, 2014. ISBN: 146149298X, 9781461492986.
- [55] *JavaScript Object Notation*. 2013. URL: <http://www.json.org/>.
- [56] *Volley library*. 2017. URL: <https://github.com/google/volley>.
- [57] *GSON: A Java serialization/deserialization library to convert Java Objects into JSON and back*. 2017. URL: <https://github.com/google/gson>.
- [58] C. Wolf and J.-M. Jolion. “Object count/Area Graphs for the Evaluation of Object Detection and Segmentation Algorithms”. In: *International Journal on Document Analysis and Recognition* 8.4 (2006), pp. 280–296.
- [59] ABBY. *Fine Reader 14*. 2017. URL: <https://www.abby.com/en-eu/finereader/tech-specs/>.
- [60] *Android Camera2Basic Sample*. 2017. URL: <https://github.com/googlesamples/android-Camera2Basic/>.
- [61] Search Engine Journal. *Google, Neven Vision and Image Recognition*. 2017. URL: <https://www.searchenginejournal.com/google-neven-vision-image-recognition/3728/>.
- [62] Johannes Bernhard Steffens et al. *Face recognition from video images*. US Patent 6,301,370. Oct. 2001.
- [63] José Luis Pech-Pacheco et al. “Diatom autofocusing in brightfield microscopy: a comparative study”. In: *Pattern Recognition, 2000. Proceedings. 15th International Conference on*. Vol. 3. IEEE. 2000, pp. 314–317.
- [64] Adrian Rosebrock. *Blur detection with OpenCV*. URL: <http://www.pyimagesearch.com/2015/09/07/blur-detection-with-opencv/>.
- [65] Hao Fang et al. “From captions to visual concepts and back”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1473–1482.
- [66] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.
- [67] Inc. CloudSight. *CloudSight API reference*. URL: <https://cloudsight.readme.io/docs/testinput>.
- [68] Xiph.Org. *Opus Interactive Audio Codec*. URL: <https://opus-codec.org>.
- [69] sephiroth74. *Android-Exif-Extended*. 2017. URL: <https://github.com/sephiroth74/Android-Exif-Extended>.

## Bibliography

- [70] Standard of Japan Electronics and Information Technology Industries Association. *Exchangable image file format for digital still cameras: Exif Version 2.2*. URL: <http://www.exif.org/Exif2-2.PDF>.
- [71] Jakob Nielsen. *Usability engineering*. Elsevier, 1994.
- [72] Google. *Overview of Google Play Services*. 2017. URL: <https://developers.google.com/android/guides/overview>.
- [73] louisyonge. *Opus for Android*. 2017. URL: [https://github.com/louisyonge/opus\\_android](https://github.com/louisyonge/opus_android).
- [74] shashir. *CloudSight API Java Client*. URL: <https://github.com/shashir/cloudsight-java-client>.
- [75] NumPy developers. *NumPy library*. 2017. URL: <http://www.numpy.org/>.
- [76] SciPy developers. *SciPy library*. 2017. URL: <http://www.scipy.org/scipylib/>.
- [77] John Hunter, Darren Dale, Eric Firing, Michael Droettboom and the Matplotlib development team. *Matplotlib*. 2017. URL: <https://matplotlib.org/>.
- [78] Peter BL Meijer. “An experimental system for auditory image representations”. In: *IEEE transactions on biomedical engineering* 39.2 (1992), pp. 112–121.
- [79] Peter BL Meijer. *The vOICe - New Frontiers in Sensory Substitution*. 2017.