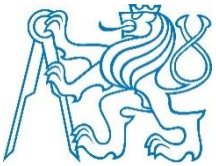


CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Department of Cybernetics

Bachelor Thesis:

AUTOMATIC COLOURING BOOK CREATION ON TOUCH DEVICES WITH OS ANDROID

Stanislav Steidl

Thesis advisor: RNDr. Daniel Průša, Ph.D.

Study Programme: Open Informatics

Specialisation: Computer and Information Science

Prague 21 May 2015

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Stanislav Steidl

Study programme: Open Informatics

Specialisation: Computer and Information Science

Title of Bachelor Project: Automatic Coloring Book Creation on Touch Devices with OS Android

Guidelines:

The task is to design and implement an application for tablets supporting work with coloring books. The application will support an automatic creation of a coloring book based on a chosen image. The target user is a child of age 3-6 years.

- Propose a method for an automatic creation of coloring books. Base it on functions for digital image processing available in OpenCV library (edge detection, segmentation).
- Analyze accuracy and reliability of the method for various types of input images.
- Utilize possibilities of touch devices. Implement the application for OS Android.
- Create a user manual and programmer documentation.

Bibliography/Sources:

[1] Šonka M., Hlaváč V., Boyle R.: Image Processing, Analysis and Machine vision, 3rd edition, Thomson Learning, Toronto, Canada, 2007.
[2] Nudelman G.: Android Design Patterns, 1st edition, Wiley, Indianapolis, USA, 2013.

Bachelor Project Supervisor: RNDr. Daniel Průša, Ph.D.

Valid until: the end of the summer semester of academic year 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 14, 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Stanislav S t e i d l

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Automatická tvorba omalovánek na dotykovém zařízení s OS Android

Pokyny pro vypracování:

Úkolem je navrhnout a naprogramovat aplikaci imitující práci s omalovánkami na tabletu. Významnou součástí aplikace bude možnost automatické tvorby omalovánky na základě vybraného obrázku/fotografie. Cílovým uživatelem je dítě ve věku 3-6 let.

- Navrhněte metodu automatické tvorby omalovánek. Jako základ metody využijte funkce pro zpracování digitálního obrazu dostupné v knihovně OpenCV (detekce hran, segmentace).
- Analyzujte úspěšnost metody podle typu vstupních obrázků.
- V návrhu vlastní aplikace plně využijte možnosti dotykového zařízení. Vše implementujte pro OS Android.
- Vytvořte uživatelskou příručku a programátorskou dokumentaci.

Seznam odborné literatury:

[1] Šonka M., Hlaváč V., Boyle R.: Image Processing, Analysis and Machine vision, 3rd edition, Thomson Learning, Toronto, Canada, 2007.
[2] Nudelman G.: Android Design Patterns, 1st edition, Wiley, Indianapolis, USA, 2013.

Vedoucí bakalářské práce: RNDr. Daniel Průša, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 14. 1. 2015

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne . . . . . .

. . . . . .

Podpis autora práce

Abstract

Main goal of this thesis is to study possibilities of implementation image processing methods, namely edge detection and segmentation. Main purpose is to automatically create colourings for children between 3 and 6 years. The result implementation is presented as an OS Android based application for tablets. The resulting application Colour it! automatically generates colouring based on provided image. The implemented detection methods are focusing on cartoon images or their photos. However the source image is not bounded just to these types, and therefore the application does generate a colouring for any type of provided image. Colour it! application also provides colouring environment for subsequent colouring. Thesis also contains evaluation of generated colourings, documentation and user manual.

Abstrakt

Cílem této práce je prozkoumat možnosti implementace metod zpracovávajících obrázky, a to jmenovitě detekci hran a segmentaci. Účelem je automatické vytváření omalovánek pro děti ve věku od 3 do 6 let. Výsledná implementace je prezentována formou aplikace pro tablety s operačním systémem Android. Výsledná aplikace Colour it! tak automaticky generuje omalovánky pro zadaný obrázek. Detekční metody se zaměřují především na komiksové obrázky či jejich fotografie. Nicméně vstupní obrázek není nijak omezen pouze na tento typ, a tudíž je možné vytvořit omalovánku pro libovolný vstup. Colour it! aplikace také podporuje následné vybarvování. Součástí bakalářské práce je i hodnocení takto vygenerovaných omalovánek, dokumentace a uživatelská příručka.

Acknowledgements

I would like to express my gratitude to my bachelor thesis supervisor RNDr. Daniel Průša Ph.D. for introducing me to computer vision, for his valuable hints and overall aid during my work which allowed me to finish this thesis. I would also like to thank to Itseez team which leads the development of open source OpenCV library for all their hard work which enabled me to work on this thesis. Last but not least, I would like to thank my family for all their support during my studies and work on this thesis.

Table of contents

Table of figures

Introduction

This chapter explains the motivation for the thesis and its aims.

Motivation

Nowadays tablets are involved in children's lives more than ever. Therefore it would be desirable to use them for activities that can improve children's abilities and skills. I have decided to implement a colouring book application.

Colouring books help children improve their creativity and mainly their fine-motor skills. There may be disagreement on whether colouring via the tablet is as efficient as classical colouring on paper or not, but I think tablets are an alternative that is worth exploring since tablets surround us wherever we go, while classical colouring books might not.

At present, there are colouring books for Android devices available. However they all lack what is - in my opinion – the most critical feature, namely the creation of new colourings by the users themselves. This opens the possibility of generating a large variety of new colouring pages which can be chosen to correspond accurately to children's needs and interests. There can be found a project which implements creation of colourings from an image [11], however, it is only a web application, so the user needs an internet connection and a browser. I did not find any application of this type on Google Play, which is the main source for Android applications, or at any other place on the internet. Therefore I have decided to create a new colouring books application which implements this feature with a fully or partially (depending on the user's decision) automated creation process from images for Android tablets on my own.

For the purpose of creating the application I have decided to implement edge detection and other image processing methods. One of main reasons why I have decided to use such methods is that I think that it is important to show that computer science is not only

an academic or high-tech research field but also one that can be introduced into everyday life, and can serve as a powerful tool even for the youngest of us.

Goal

The goal of this work is an application which will provide automatically customized methods based on an initial image. These methods should be used to create accurate and simple colouring pages. Since there are not any universal object detection methods and it is still an unsolved problem, this application will focus on detecting in specified areas of images. The application should be able to handle simple cartoon images flawlessly. It should also handle the import of prepared colouring pages or detection over average cartoon images or simple photos, such as a photo or image from a fairy tale book. Since the idea is to use a click-to-fill method of colouring, a gap in an edge could mean a complete failure of detection. Therefore in the thesis I will not try to solve complicated photos which pose problems for detection and would require a lot of computational time, which is still limited on such devices.

The program should also provide an environment for experienced users to manually set variables to further enhance the results of detection. Another very important part is to provide a colouring environment for comfortable colouring which will provide template display, click-to-fill colouring, and a unique colour set for each colouring page corresponding to the original image, but also the export of prepared colourings for possible print and colouring with real pencils afterwards.

Theoretical Background

This chapter describes the theoretical background which is necessary for this work. However, it is expected that the reader is familiar with the basics of Java, the least square metric or gradients.

Android overview

Android is a Linux based operation system released by Google in 2009. Based on Google statistics: “Every day more than 1 million new Android devices are activated worldwide. “[7] It is estimated that nowadays there are more than 85% of Android devices with OS version 4.0 or newer. Based on Digital Trends statistics [16] from the end of year 2014, it is also the most used operating system in mobile devices and its market share is still growing.

Operating system	% share of global OS market
Android	81.5%
iOS	14.8%
Windows	2.7%
Blackberry	0.4%

Programing for Android is supported in the Java language Android library which contains all necessary API to create all kinds of applications for Android devices. The programs for Android devices are therefore written in the Java language. Since it is expected that the reader is familiar with Java basics, the main differences between Android and Java programming will be explained here. In Android programming there is no Main function. Instead of Main function there is a Main Activity which will launch after starting the application. Every Android application is composed as a set of Activities with only one active Activity which is displayed at the device screen and with the rest in idle or sleep mode. The active Activity can be switched to another while the previous one is put to sleep.

The graphical layout of each Activity is described by an XML file which is the same for many other constant resources needed for the application. Event handling is similar to Java: it is performed by adding event listeners.

For the purpose of detecting gestures on the display there is a library prepared which invokes one of events whenever it detects one of standard gestures. This is useful for zooming an image. There is also another library prepared for getting coordinates of clicking the display which will be used for detecting the correct surface to be coloured or picking the colour.

For every Android application you can set its behaviour in landscape and portrait orientation. This application is locked in landscape mode for better usage of the screen, which is already too small in many cases. That is also the reason why this application is recommended only for devices with the diagonal 7” and greater. This practically eliminates all smartphones and leaves tablets only. The reason is that colouring on a smaller device would lead to unpleasant results where the colouring page would be too small to be coloured properly. However, this is only a recommendation because it is not possible to forbid installation on devices which do not meet this requirement.

Every Android application has a minimal and a target version of the Android system. The minimal version is the lowest possible Android version required to install that application. Since image processing methods (which will be described later) need quite a lot of computational time, I have decided to set the minimal Android version to 4.0, which should guarantee a sufficient amount of computational power.

The Android library also provides a large set for working with Bitmap which is the main object class for images in Android. Therefore there are also static methods available for working with them, such as creating of new Bitmaps or changing target pixels in others. Since the Bitmap is the default and also well-designed class for images, it is used for holding and any operations with images in this work. Since the Android Java package does not contain any methods for computer vision, an additional library will be necessary.

OpenCV

As noted in the previous chapter, the Android library does not include any tools for image processing. Therefore it is necessary to use some external library which will fill up this gap. Several computer vision libraries are available (OpenCV, javaCV, imageJ, boofCV), yet not all of them offer precisely what will be needed. OpenCV is indisputably the most robust, most commonly used and most optimized out of all available BSD licenced libraries, and it also has a well accessible documentation. Therefore the OpenCV library for Android is being used in this work. OpenCV library is a widely used computer vision library, which was initially written in C++ language and released in 2000. It is developed by Itseez team. Full interfaces for Python, C, Matlab as well as for Java are now available. OpenCV is a large library which covers a wide area of uses, such as image processing, motion tracking or statistical machine learning. For this work we will use only image processing, namely Canny Edge Detection, Gaussian Blur, K-means colour quantization and thresholding. A description of these methods can be found in the following chapters bellow.

Even though there is a Java interface, the library is composed as a C++ library, which is why the whole interface has more in common with C++ than with Java code structure. This might be quite confusing especially for those who are not familiar with C++. OpenCV provides a full API description on their website [6]. However, it is often very brief and sometimes rather confusing.

For Android devices it is recommended to install OpenCV library as a self-standing application rather than include the library into applications directly. This has multiple reasons but the main is that OpenCV library can utilize the multiple-core architecture which is very common in Android devices better. Another important reason is that OpenCV is quite a large library – it takes about 20MB – so the developer should make sure that the user has it installed only once on his device, rather than having it installed for every single application which benefits from OpenCV.

OpenCV uses Mat object as the basic entity. Mat object consists of channels (one for each RGB colour) which are represented as matrices. However, the accessibility to a single pixel is very poor and expensive in terms of computation. On the other hand, Mat objects provide a great support for matrix multiplications and other matrix operations. OpenCV also supports

conversion from Android Bitmap object, which is the basic Android image format, to OpenCV Mat object in both ways.

K-means clustering

K-means clustering [1][402] is a method which divides the data set into K clusters where every single data vector (in this case the coloured pixel, the 3-dimensional colour vector RGB) is aligned to the nearest cluster. The method consists of an iteration over two steps after initiation. In the initiation the starting centres are selected based on a chosen strategy. The most common strategy is a random choice of K vectors from the dataset, which obviously does not guarantee any kind of optimality, and it has been shown [15] that in the worst case the running time can be even super-polynomial. This is why in this thesis a K-means++ method has been used to choose initial centres. The initiation is followed by an assignment step. In this step, the nearest cluster-centres are found for all data vectors, so the data set is divided into K clusters (The nearest cluster-centre is defined as the one with the lowest least-square distance.) When the evaluation step is completed the next step is the update of centres. For each cluster a new centre position is chosen. It is chosen to minimize the sum of distances of all vectors from that cluster (distance is defined as least-square distance). After this we repeat the assignment step and the evaluation step, respectively, until no cluster-centre changes its position. This method will not guarantee the global optimum because it converges to local optimum only. The optimality of the result is strongly dependent on the initial distribution of cluster centres. That is the reason why K-means++ initial distribution is used here because it guarantees upper bound of optimality as (log k) competitive to optimal solution. Since OpenCV supports K-means++ no additional implementation is needed**.**

Figure 1 gradient based colour mixture

In this work the K-means clustering has been used as the colour quantization method, which transforms an image into an image containing only K different colours. These K different colours are cluster centres gained from the K-means method. This simplification of an image is useful in edge detection in areas where colour is slowly transforming from one colour to another.

Figure 2 3-colour quantization example

Whenever K means is used, there is one potential problem. And it is a question how to select a correct number of clusters. This is very subjective question and therefore there is no general answer. There are some state of art approaches like the elbow method and others [17]. However they are all too computationally complex and this work requires a quicker approach. Therefore the K is determined as:

$$K = max\left\{k; \frac{c_{solid}}{10}\right\}$$

Where $k = 15$ has been set based on observation of testing images and c_{solid} computation is based on analysed image and is further described in Implementation chapter.

Another use of this method is to generate a corresponding colour palette for future colouring, which makes the colouring much more similar to the original.

Thresholding

Thresholding is a simple method used on grayscale images which performs an action based on threshold value. In this case binary thresholding has been used. In binary thresholding each pixel is compared to a given threshold and if the pixel value is smaller than the threshold then it is set to 0, while if it is greater or equal then it is set to 255. This corresponds to black or white colours. The method is very useful in cases where prepared colourings were downloaded to determine exactly where the areas are. This is necessary especially when anti-

aliasing was previously used on the image. Anti-aliasing is a method employed to smooth the image so that the contours look more fluent. Binary thresholding can be used for reversing an anti-aliased image to an aliased one.

In the OpenCV library this method is also used to count Otsu's threshold, which is useful in edge detection.

Canny edge detection

Edge detection [1][144] is a name for a set of methods which use brightness changes in the image to detect the so-called edges. There are many approaches to the detection of edges. In this work, state of art Canny Edge Detection has been chosen.

Canny edge detection consists of multiple steps. After filtering noise from the image, for example by a Gaussian filter, the next step is finding the brightness gradients. This is done by the Sobel procedure, which will not be explained here. When we have the gradients, a non-maximum step is performed. This step removes single pixels which had been detected as potential edges but stand alone. Since the method aims at detecting edges which are basically curves, there is no point in keeping single 'dots'. After this step, only thin lines or curves will remain. The last step of Canny is Hysteresis. In this step, only those edges which are between the lower and upper threshold are accepted. These thresholds are given to the Canny detector as an input alongside the inspected image.

Otsu's threshold

Otsu's threshold is the best binary threshold which divides the target image with the least possible variance into two sets.

State of the Art

This chapter describes the solutions currently available for the colouring task.

There are many other colouring applications at Google Play store. For a comparison I have chosen 2 applications: Toddler coloring book free/pro and Coloring book. Considering the number of total downloads and user ratings, both are interesting from different points of view. For the purposes of comparison I have also included a web application Scrap coloring. Since it is not an Android application full comparison is not possible, however, it is the only solution I have been able to find which implements automatic creation of colouring pages from images.

Toddler Coloring book free/pro

Figure 3 toddler coloring book icon

Toddler coloring book (TCB) has a very lightweight approach which is based on very simple colouring of a few included colouring pages.

TCB is available in a free or paid version. The paid version is an extended version of the free one, providing a larger pool of predefined pages. The free version has 8 pages, and the paid one 58. It is a well-rated application with 4.6 stars of 5 based on rating of more than 43 thousand people. It has over 1 million of total downloads.

This application, as the name hints, is targeted at 1 to 3 year-old children. Therefore the interface consists only of a colour picker and the colouring itself. There are no menus or advanced settings. The buttons of colour picking are big enough for even a toddler to use without the support of any adult.

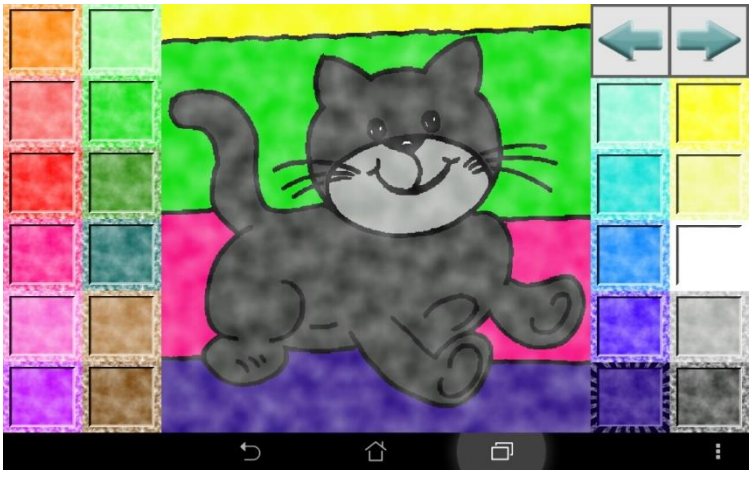


Figure 4 toddler coloring book example: cat

Coloring book

Figure 5 Coloring book icon

Coloring book (CB) is an application with over 14 million of total downloads. It is rated as 3.8 stars of 5, based on over 38 thousand votes.

CB consists of more than 400 prepared colouring pages sorted into groups based on colouring difficulty and theme. It offers a large scale of degrees of difficulty, which makes the target customer category wider.

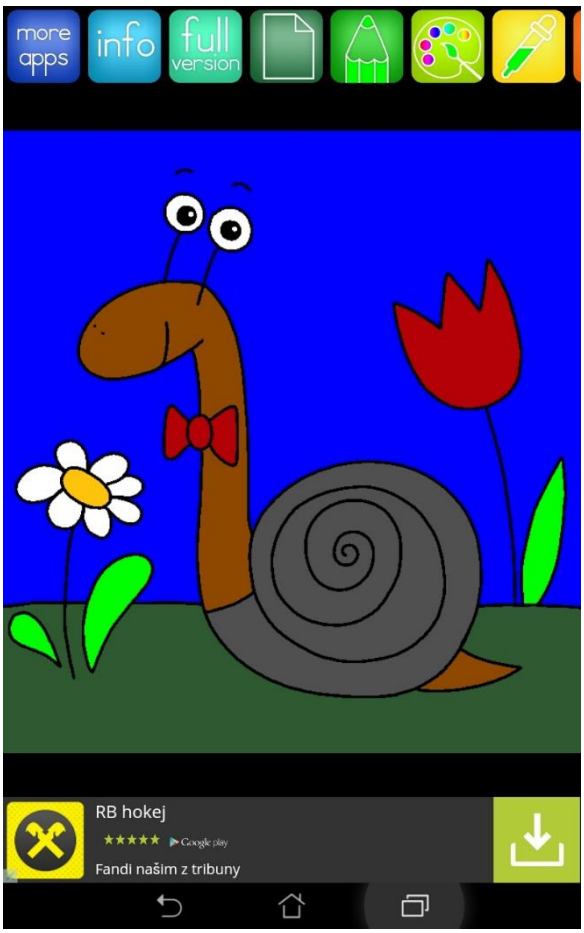


Figure 7 Coloring book example: snail

Figure 6 example: dwarfs

The colouring interface is rather complicated. Functions are available in a menu which is positioned at the top of the screen and is composed of simple images. However, it can be confusing since each menu item has a different colour and a toddler may assume, for example, that it is not a menu but a colour picker itself. The function for the picking of colour is hidden

behind the pencil, the palette or the yellow button, depending on the user's preferences because each offers slightly different possibilities. Since performing the basic function of picking the colour requires multiple touches of the right places, the target customer category is definitely not toddlers and is not suitable for pre-school children either.

On the other hand, CB offers multiple advanced functions, such as the zoom, which is almost mandatory to use when colouring hard images, or the undo function which reverses the last action the user made. Another advanced function is exporting the colouring for the purposes of sharing or printing.

A paid version is also available but it is not so well-rated or downloaded. In the paid version, there is an interesting feature of importing prepared colouring pages.

Scrap coloring

Figure 8 Scrap Coloring icon

Scrap coloring (SC) is a webpage focused on colouring. Since it is a webpage, it can be used only with internet connection. It contains a database of prepared colourings for online colouring or printing, but also a section for creating new colourings from the pictures provided by the user. The results of this feature are largely dependent on the input image. An example of one well-made and one not so much well-made colouring from a provided image is below.

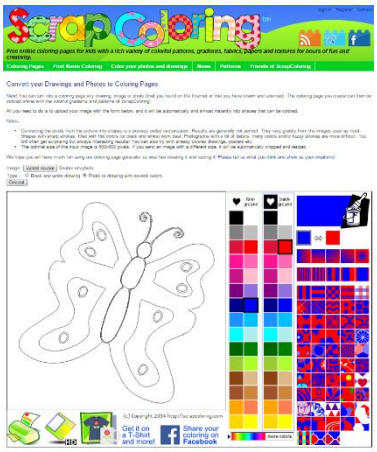


Figure 9 Scrap Coloring detection

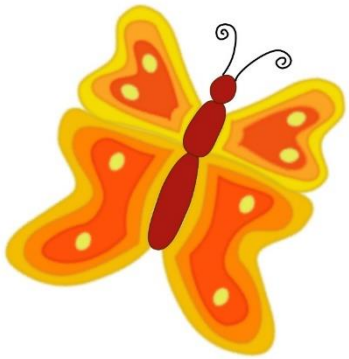
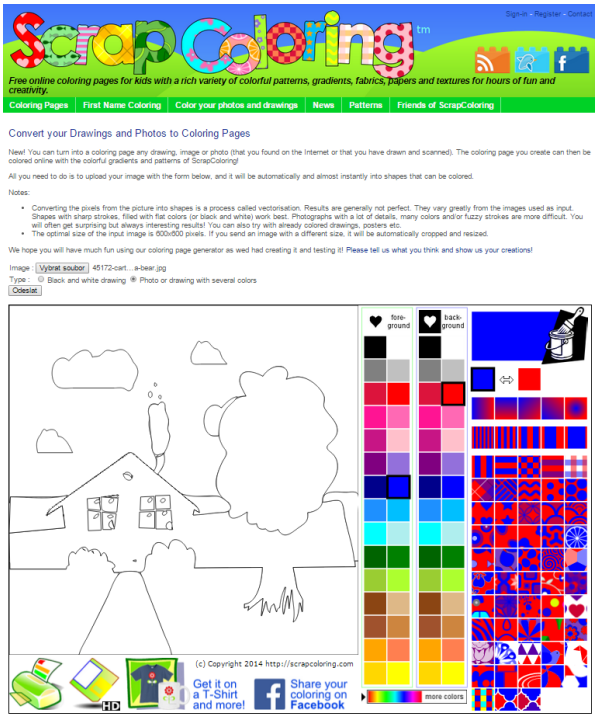


Figure 10 source butterfly



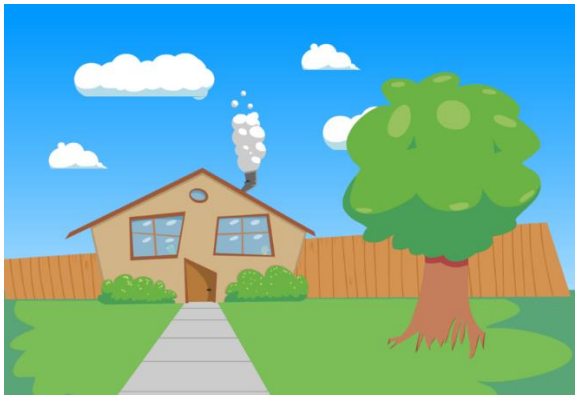


Figure 11 source house

Figure 12 Scrap Coloring detection: house

Since the SC colouring environment is a webpage it can also be used from Android devices. It does not require flash or any other plugins which usually cause problems on the Android system. SC offers a large variety of colours and for experienced users there is also an implementation of gradient mixing of two colours or the use of a texture.

Export of the result is also supported. It is possible to print or download it, however, the resolution is always 600x600 pixels, which might be sometimes insufficient and may modify the picture which was originally wider or taller.

The colouring environment is simple and easy to understand. Nevertheless, the creation of a colouring is more complicated and would require someone skilled enough to provide the source image.

Comparison

I have described three approaches to colouring tasks, and now I will summarize the advantages or disadvantages of these solutions and compare them.

Toddler coloring book

The colouring interface is simple and easy to understand. However, there are not many pages available and the toddler is likely to find colouring the same page over and over rather boring.

Conclusion

This solution aims at similar users as my work. However, the functions it provides are very simple and it has a very limited pool of pages.

Coloring book

The database of pages is large and there are pages for everyone. Page selection is well-organized into difficulty groups which are divided into smaller groups on the basis of the themes they present. Page selection is simple and does not require an experienced user. On the other hand, the main menu has too many items and performing actions which are repeated many times during colouring, such as picking a different colour, requires too much clicking and the actions are harder to find than they should be.

Conclusion

This solution is nice, with plenty of pages and advanced functions, but it aims at older users. At the same time, some key functions, such as the creation of new pages, are missing.

Scrap coloring

This browser based implementation has a lot of disadvantages. Most importantly, the user needs internet connection and colouring from Android devices is slower than it should be. On the other hand, the creation of unique colouring pages based on user-provided images provides an inexhaustible source of new fresh colouring pages. Some may not be perfect though.

Conclusion

This solution offers suitable colouring environment and all key functions, even though the creation of pages may not be perfect. However, a fundamental disadvantage is the need for internet connection because the user has to wait for the server response. This can take several seconds (depending on the connection) after each action, which can be considered quite slow.

Recommendation

For this work I have decided to combine the design philosophies of the above applications in order to achieve a better result.

The most common user interface philosophy is implemented in Toddler coloring book. Since our target group is pre-school children the colouring environment has to be kept simple.

As far as choosing right page is concerned, the best approach is represented by the Coloring book application. Sliding trough previews of pages seems best in terms of using the potential of the touch display.

The approach of selecting source image and letting the rest on device seems to be the only way for this work's target age category. However there is a potential problem, since the selection of new image cannot be implemented in simpler way and it still might be too complicated for some young users. Therefore we assume that there might be a second, more experienced user (a parent, for example) who will help the child to create new pages. With the experienced user in mind the work should offer advanced creation options beside the basic ones, which would let the user influence the process to achieve even better results.

Implementation

In this chapter we discuss the implementation of more sophisticated methods which are employed in creating the colourings and their colouring afterwards. Further details of implementation can be found in Documentation chapter bellow.

Detection implementation

The image format covers a large field of possible source inputs. It can be a high-definition photo or a very simple image drawn for example in Microsoft Paint in Windows. Realizing this fact leads to the conclusion that it would be wise to use different approaches for different kinds of source images. Therefore the implementation can be divided into two main steps. In the first step (Source recognition) the metrics are applied to determine what kind of image the algorithm is dealing with. In the second step, the knowledge from the previous step is used to modify detection according to the source image.

Image classes

Basic classes of source images can be determined based on the behaviour of detection algorithms. These classes react similarly and yield similar results based on same settings. It is important to mention that human perception does not work the same way as computer based edge detection. Therefore images are not divided into those classes by human perception but by mathematical metrics.

Simple cartoons

This is the most common class for this work target group. Such an image consists of blocks of solid colours. The class includes, for example, templates of colourings or simple pictures without advanced shading.

Figure 13 an example of simple cartoon class

Advanced cartoons

This is a more advanced class which contains images with more numerous and smaller surfaces of solid colours than simple cartoons. For example, there can be gradient colour mixtures or lighting effects.

Figure 14 an example of advanced cartoon class

Photos of cartoons

This is a class of photos of cartoons which had been scanned or photographed previously. Some noise is present in these images; however, there are still clear surfaces even though they are not made of solid colours.

Figure 15 an example of class containing photos of cartoons

Real photos

This is the most general class where a lot of noise with no clear surfaces is expected. Therefore the result is largely dependent on the source image, its quality and content.

Figure 16 an example of photo class

Colourings

This class comprises downloaded colourings prepared to be coloured. Images are grayscale and consist of white and black-like colour. Only real issue is that black colour is often shade of grey and not strictly black.

Figure 17 an example of colouring class

Source recognition

As an input to this step a resized image of unknown content is expected. At the end we expect a set of settings for a detection algorithm. Since the content, density, source or almost any other information about the source image are unknown all assumptions are based on the current state of the image. Therefore the only source of information is a matrix of pixels where each pixel has an integer value which describes the colour it presents.

First step is to compute the Otsu's threshold (OT). The OT is used in two ways. First it initializes Canny edge detection, and second it determines if the source image is a prepared colouring or not by the method described in chapter – Colouring class detection - below. After this sub-step, it is clear whether to continue or to skip the rest of source recognition.

The most straightforward metrics is counting the size of the set which contains all unique colours and their shades in the source image. This metrics (Colour count) is good to determine the difficulty of the source image. Based on observation, photos usually contain approximately over 80 000 different shades due to noise and, on the other hand, simple cartoon images usually have less than 10 000 shades. It is a large number especially for simple cartoons which optically consist of just a few colours. Most of those colour shades are found at edges. This is caused by anti-aliasing techniques which are applied for example during image reshaping. Second useful metric is counting of solid colours.

A Solid colour is defined as RGB colour such is contained at least 100 times in analysed image. Solid colour metric helps to distinguish the correct class. It gives a different point of view about the image. Simple cartoon class simply does not contain that much colours. On the other hand, blurred images (which usually comes from photo classes) does contain fewer solid colours and more colour shades. Therefore combination of those two metric can tell whenever to use noise reduction techniques and when it is probably pointless. Solid colour count is also used in computing how many clusters will be used in Colour quantization.

Image class	Average colour shades	Average solid colours
Simple cartoon	Less than 20 000	Less than 100
Advanced cartoon	Less than 60 000	More than 100
Photos of cartoon	6 000 to 150 000	Less than 350
General photos	60 000 and more	350 and more

Dilatation

Dilatation is a method of thickening edges. It is useful in connecting edges which are close to be connected but there is still a gap between them. It can be also used to 'underline' core edges. In this work the dilatation is used with a cross element. The cross element is iteratively placed to every pixel of the edge and every pixel the cross element covers is set to be an edge after dilatation.

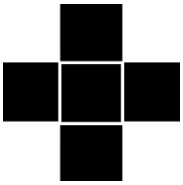


Figure 18 cross element

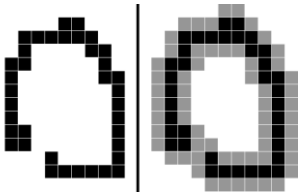


Figure 19 dilatation example, the cross element has been used

Prolong edges

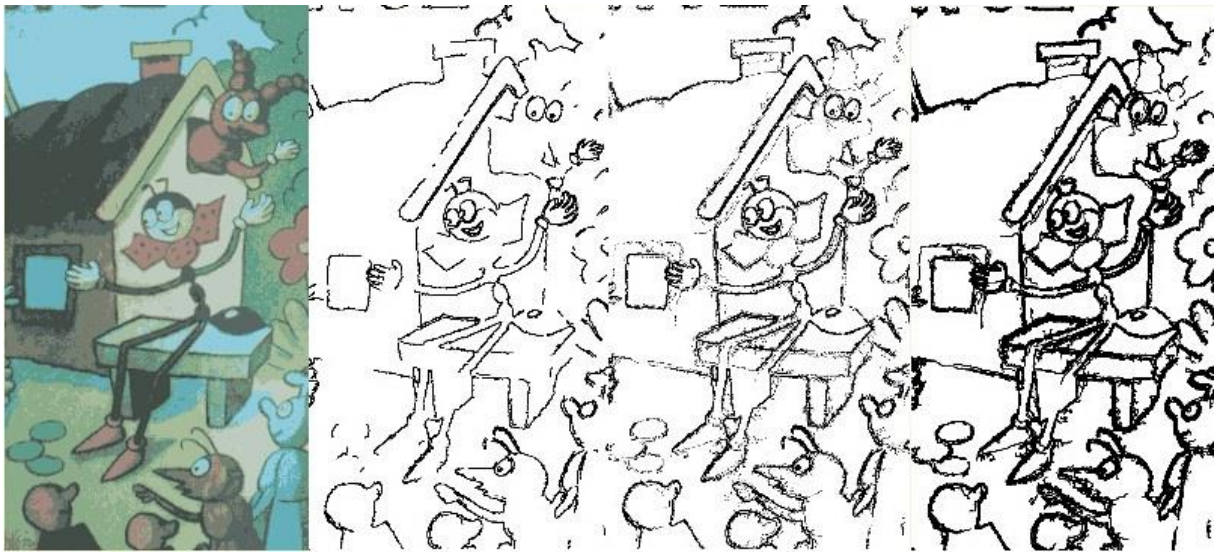


Figure 20 example of prolonging - From the left side: 1] initial image 2] initial detection 3] prolonging method had been applied 4] dilatation to close gaps

Prolonging of core edges is a method which allows to detect edges in noisy images. The idea is to find cores of real edges with strict thresholds and iteratively prolong them.

At the initiation the method sets some strict threshold which will produce just core pieces of edges. With each iteration the threshold bounds are loosened a bit. One iteration consist of the following steps:

1) Loosen the thresholds a bit.
2) Detect new potential edges with the loosened threshold.
3) If any of potential edges are connected to real edges from previous iterations, append such edges to real edges.
4) If a sufficient number of new edges has been found continue with a new iteration, else terminate it.

The main advantage of this approach against a simple edge detection is that it allows to find weaker parts of edges which would be otherwise covered with noise. A critical part of prolonging is how to set correct initial thresholds. In this work they are computed based on Otsu's threshold.

Colouring class detection

Otsu's threshold (OT) is a division marker which indicates which pixels of a grayscale image will be black and which will be white. By comparing the original source image with the one created by thresholding it can be said how much it has changed. Based on that, it can be determined if the source image was meant to be a colouring or just an ordinary grayscale image.

The comparison of the source and thresholded image is performed by computing over white pixels only. Based on what we know about thresholding it can be said (and easily proved) that:

$$W_{source} \leq W_{thresholded}$$

Where *W* determines quantity of white pixels in respective image. The decision whether image is a colouring (1) or is not (0) is computed by following function:

$$d(W_s; W_t) = \begin{cases} 0 & W_s = 0 \\ 1 & \frac{W_t}{W_s} < c \\ 0 & else \end{cases}$$

Where $c = 1.2$ is experimentally gained constant.

Simple cartoons detection

For simple images it is not necessary to choose a difficult approach. It is specific of this class that there is almost no noise present in the image. Therefore the only real danger of using mild thresholds does not apply. In combination with dilatation the resulting colourings are clear and the edges are as complete as they were in the original image.

Advanced cartoons detection

When dealing with colour gradients mixtures, the Canny edge detection has a weak spot. Therefore it is necessary to use colour quantization before Canny can be used. Colour quantization will separate the gradient colours into two or more solid surfaces which can be easily detected afterwards. After colour quantization the image is very similar to a simple cartoon from the Canny detector point of view. However, since the image is more complicated, better results are achieved if edge prolonging is applied.

Photos of cartoons detection

The result of this class is strongly dependent on the source. Since photos, especially those from mobile devices are often motion blurred and noisy it is hard to deal with them. There are methods which can detect and slightly revert the motion blur but there is not sufficient computational power for such processes. Resources of Android devices are often limited and the user will not wait minutes for a single colouring. In order to normalize noise the Gaussian blur is applied. Afterwards the noise-normalized image is colour quantized. This simplified image is less noisy while core contours are still left strong, or at least as strong as they originally were. The edge prolonging method is applied to these simplified images. During the application of the prolonging method a dilatation method is injected several times. The number of injections is based on the source image difficulty and varies from 1 to 3. The Injection of the dilatation method into the process of prolonging aims at connecting some edges which might be moved by motion blur, and may consequently be left unused since they are not actually connected to the core parts of edges and are considered as noise. Simplification methods may not be strong enough in some cases and noise can crawl into real edges. Therefore a breaker of prolonging has been set which stops the iteration if over 25% of image is covered with edges.

Customized creation of colourings

Since human perception is stronger in many ways than the computer, it would be wise to allow the user to use his advantage of additional knowledge to improve the colouring creation. Therefore an interface is implemented which allows the user to employ detection techniques but also set the parameters manually. The toolset consists of Canny with settable thresholds, K-means with settable number of clusters, applying single iteration of edge prolonging and edge thickening performed by dilatation. Results achieved by an experienced user may differ in the more difficult classes but in the simple cartoon class it should not be possible to achieve a better result.

Getting palette

The idea of getting a colour palette based on the source image is fine, however we should not forget that there is not only one correct way of colouring. Therefore it is wise to add a static part of the palette which will provide all missing colours that might not be present in the original image so the user could make his own choice on the basis of his feelings rather than being limited by the application.

Getting the static part is simple, because it comprises a predefined list of colours. Getting the customized part is more complicated. It is implemented by getting cluster centres from colour quantization method K-means. This method has been chosen because this set describes the image best from a mathematical point of view. It is also important to note that colours contained in the palette may not even be present in the original image.

Results

This chapter describes the results of the implementation of the methods which were dealt with in the previous chapter. The results bellow are divided into classes which are described in previous chapter.

It is very important to note that the results of detection are very much dependent on the source image. If a user chooses an image which would be difficult for him to transform even manually, then it cannot be expected that the result of this detection would do any better.

All the results were made as screen captures of the testing device [18]. The example testing sets of images consists of 5 images from each class, which had been chosen to cover all possible issues of the specific classes.

Classification

The results were evaluated based on the accuracy of detection. Evaluation must also consider the purpose of detection. That means that areas must be well-bounded without gaps. Evaluation may not be precise since the target group of users is not capable of rating properly.

Evaluation is performed by rating from 0 to 5 stars, which is same system as the one used for rating applications in Google Play Store. Star rating is accompanied by a brief explanation of rating.

Stars are gained for fulfilling the following criteria:

value	criteria
0-2 stars	Main object detection without gaps
0-1 stars	Background detection without gaps
0-0.5 stars	Noise filtration
0-0.5 stars	Corresponding amount of edges
0-0.5 stars	Corresponding edge thickness
0-0.5 stars	Overall appearance

Detection Evaluation

Simple cartoons

Simple cartoons are easiest for detection and therefore evaluation can be stricter than in more difficult classes. What is expected is a gapless detection with possible issues of unnecessarily thick bounds.

Bouquet

4 of 5 stars. Main edges are thick. Inner edges are thinner and describe the details inside well enough. Only the red flower in the middle could be described better.

Figure 21 detection example

Fish

5 of 5 stars. No surface is forgotten, no gaps. Edges are not thicker than necessary.

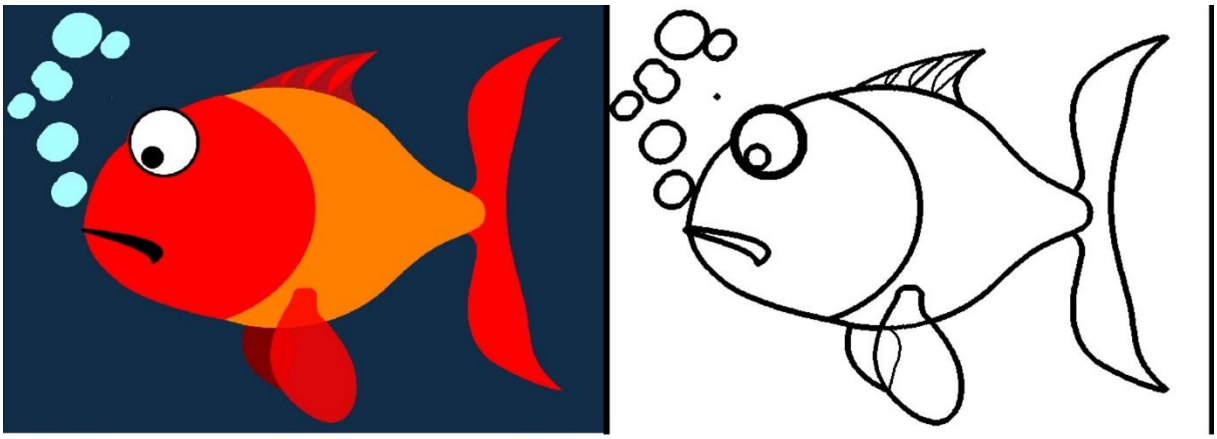


Figure 22 detection example

Tractor

4 of 5 stars. Centres of wheels could be described better. Wavy lines are caused by low resolution of input image and therefore this defect should not have any impact on the evaluation.

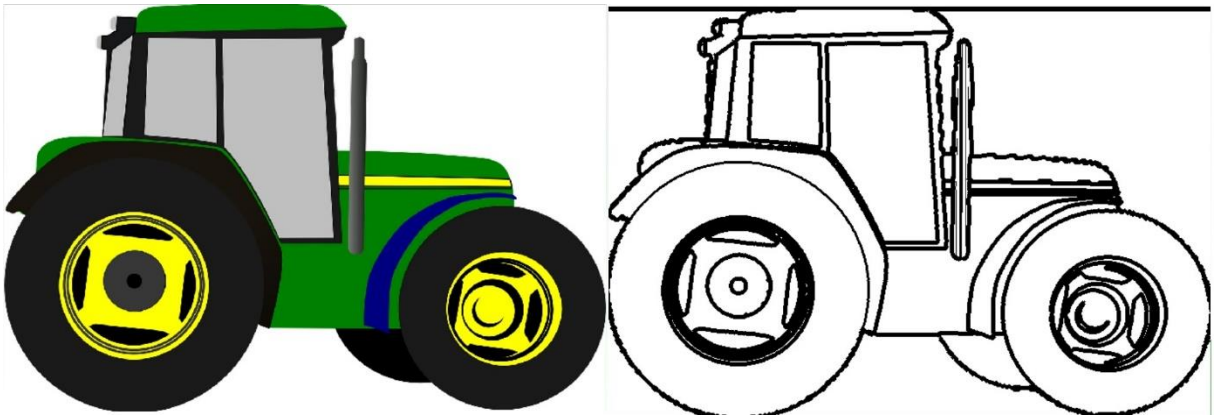


Figure 23 detection example

Mole

4.5 of 5 stars. The source image is simple but contains lot of noise due to low resolution. However, false edges which are visible on the watering can should not be present.

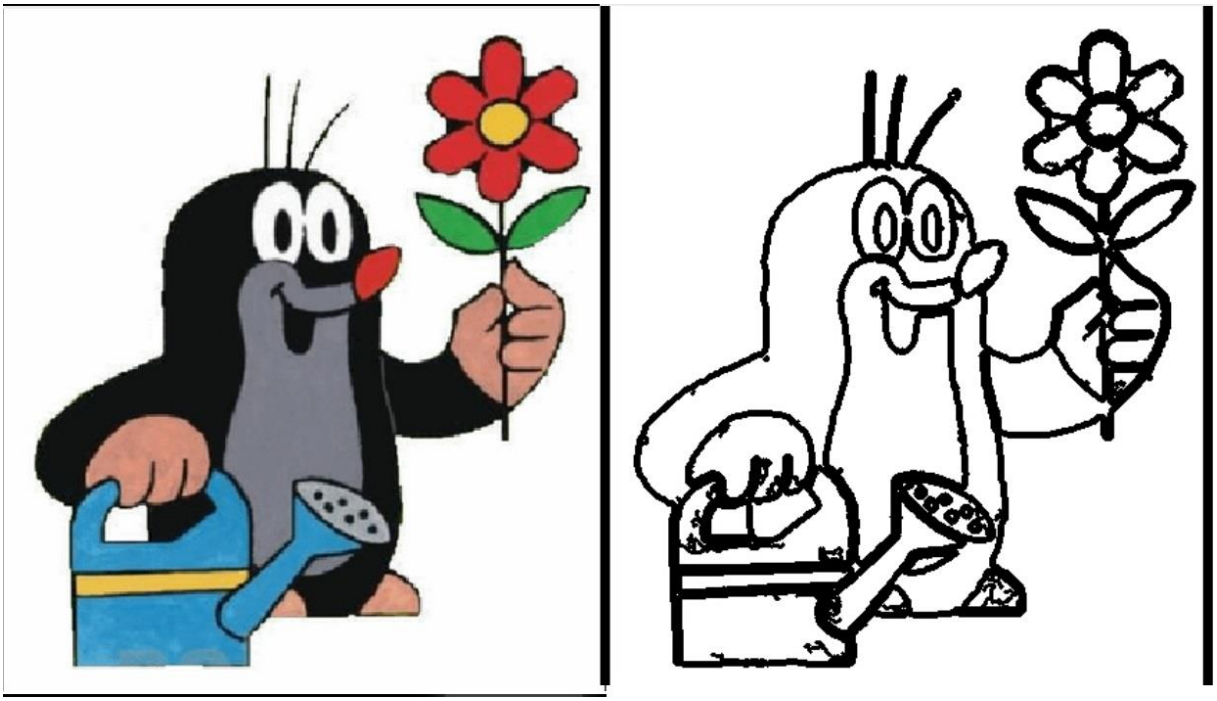


Figure 24 detection example

Helicopter

3 of 5 stars. Edges are pointlessly thick. However, the overall description of the image is good and the surfaces are cleanly separated.

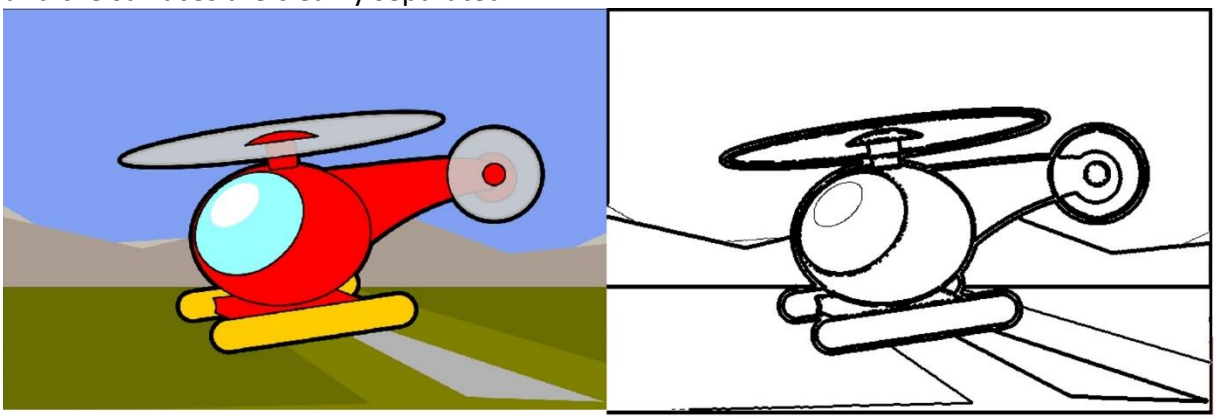


Figure 25 detection example

Advanced cartoons

Advanced cartoon class is slightly more dependent on provided original image. Therefore it is not expected to see results of such quality as in Simple cartoon class.

Lion cub

4 of 5 stars. The trunk of the baobab is not separated from the ground. Good detection of the lion cub where there could be a problem with shading on his chest.

Figure 26 detection example

Flowers

3 of 5 stars. The purple flower is not well detected. The rest of the image is well detected but the gradient colour-mixture was simply too big for smoothing methods.

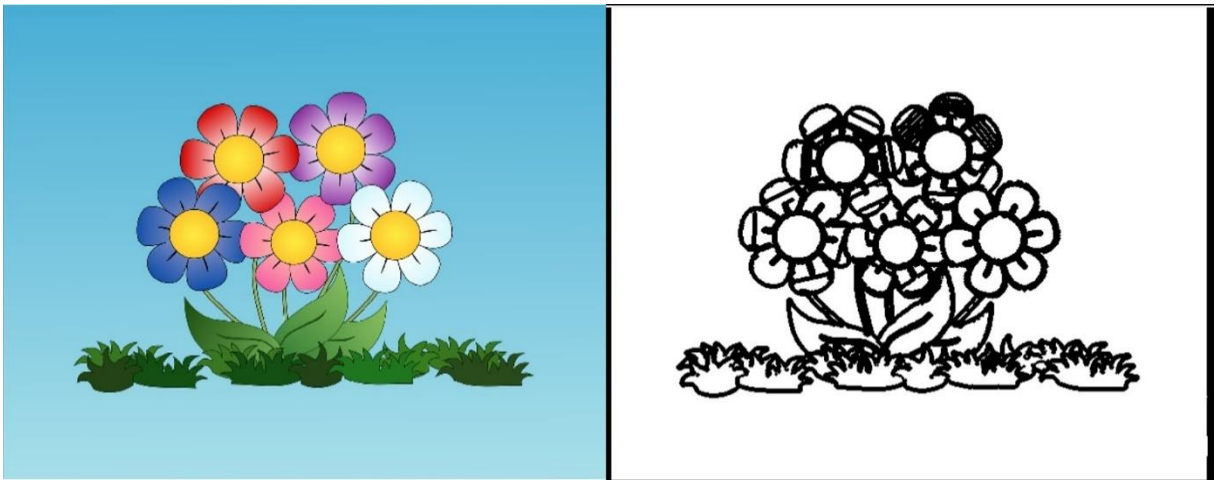


Figure 27 detection example

House

5 of 5 stars. All surfaces are well recognized and details of the drawing are carried as thin lines without making the colouring too messy.

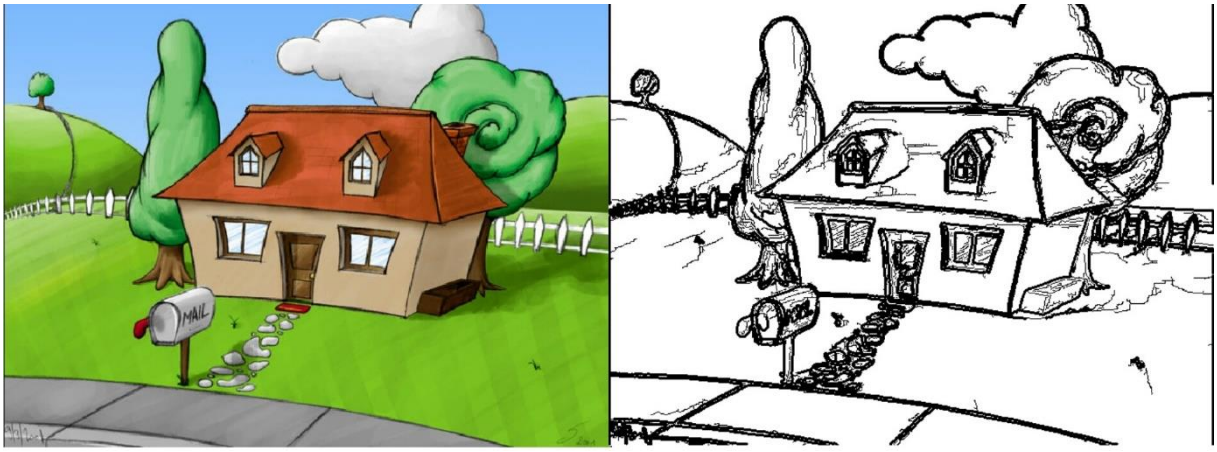


Figure 28 detection example

Animals

4.5 of 5 stars. Hard source image with lots of details, probably containing too many details for the target group. From the detection point of view there is a problem with the rock in the left bottom corner of the image. There are too many edges present which do not correspond to the source image. However, this problematic spot is marginal since the important parts of the image are the animals rather than the rock.

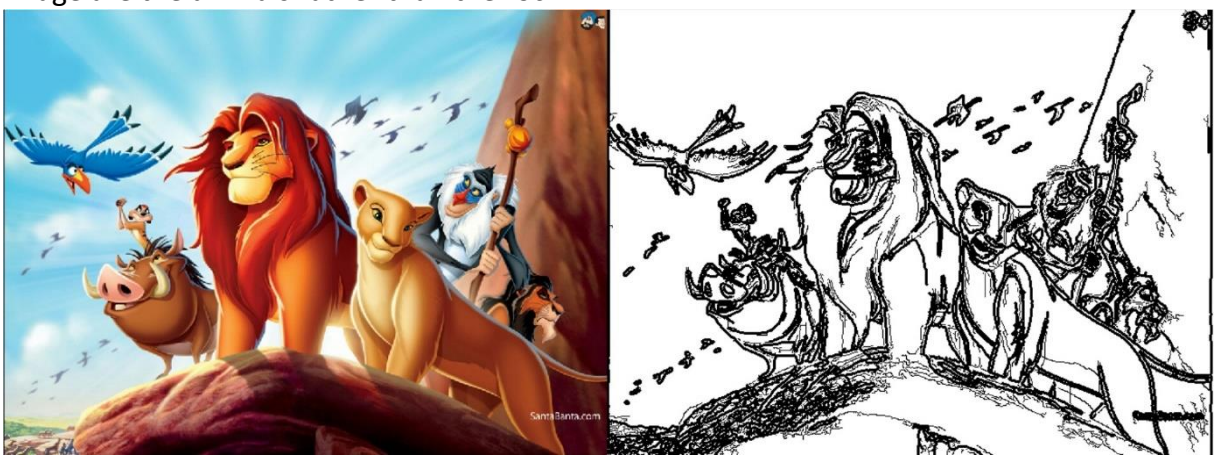


Figure 29 detection example

Duck

5 of 5 stars. The detection is flawless. Grass is detected as thin lines, which describes the image even better.

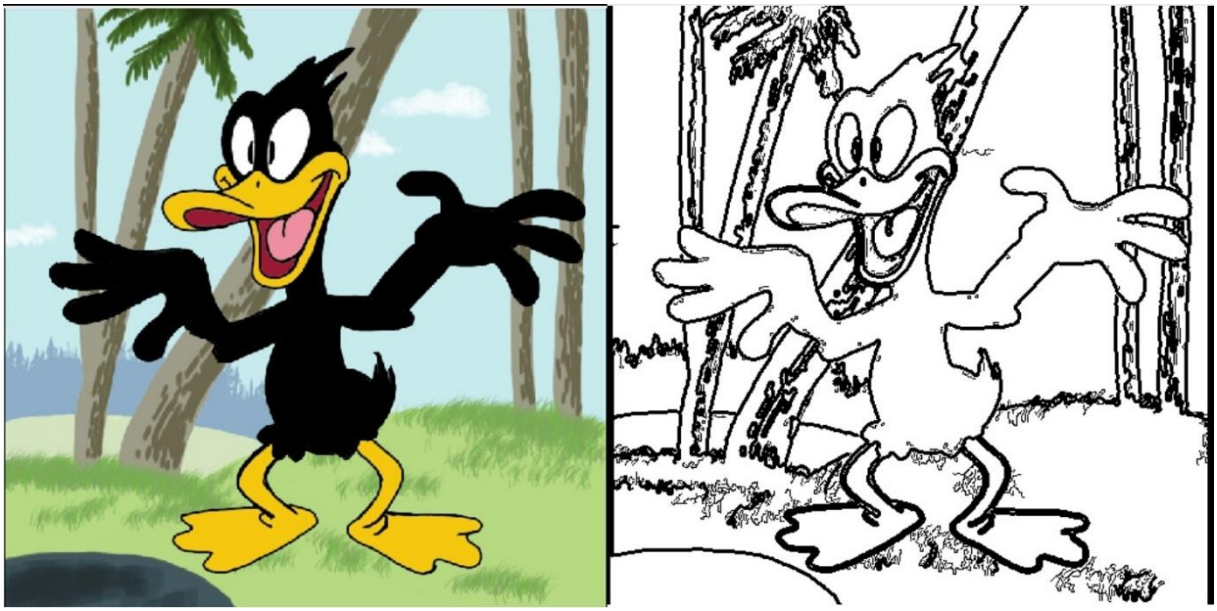


Figure 30 detection example

Photos of cartoons

Photos were taken by the test device's camera [18]. The camera has a 2Mpix resolution. For purpose of demonstration, the illustrations from children's books [13], [14] were chosen as objects of photos.

Flower

5 of 5 stars. Flawless detection of a simple image.

Figure 31 detection example

Lizard

4.5 of 5 stars. Well-detected complicated colouring.

Figure 32 detection example

Hound with an umbrella

3.5 of 5 stars. Water drops were not detected. The bounds should be thinner. However, the overall impression is great.

Figure 33 detection example

Snail-carriage

4 of 5 stars. The wheel of the carriage is not separated from the carriage itself.

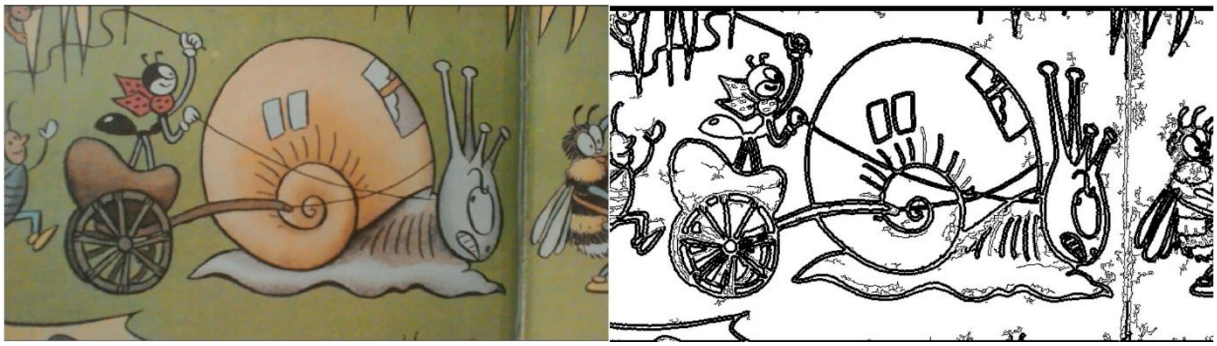


Figure 34 detection example

Bugs on a mushroom

2 of 5 stars. Missing white discs, the right shard of the bug on the left is not separated from the mushroom.

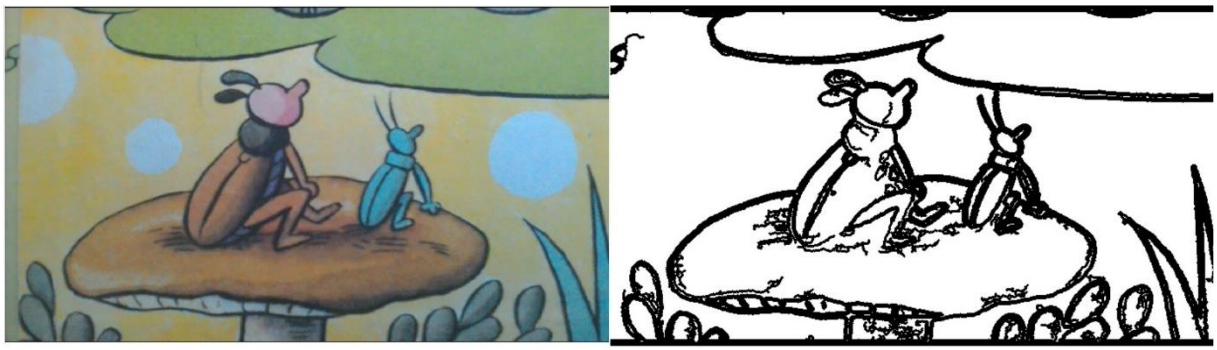


Figure 35 detection example

Real photos

Unlike previous classes this one is not a target source. Therefore these results cannot be taken as seriously as the results from previous tests.

A typical successful result from this class is divided into more surfaces which are smaller. The typical result colouring is often too complicated for target users, however colouring is not impossible.

Norwegian church

3.5 of 5 stars. Even though there is a lot of needless edges, the overall appearance is good. The main surfaces are well-bounded even though they may be composed of multiple smaller pieces. This might be confusing for the target group but a more experienced user should be able to deal with it.

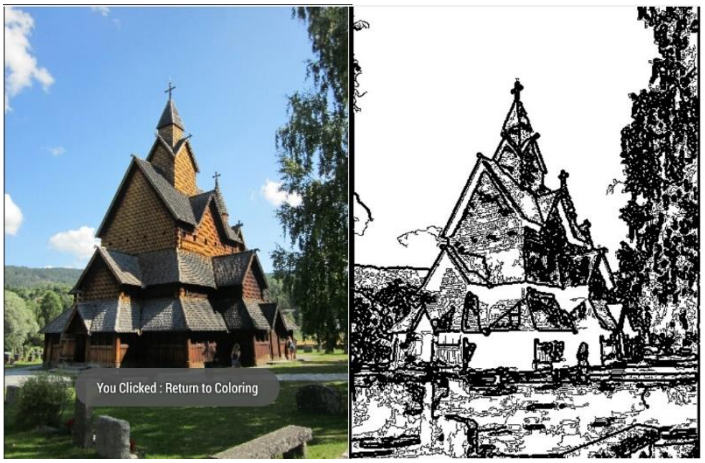


Figure 36 detection example

Water Lilly

3.5 of 5 stars. The colouring contains too much noise, which makes the colouring very difficult. However, the main object is clear enough to be coloured.

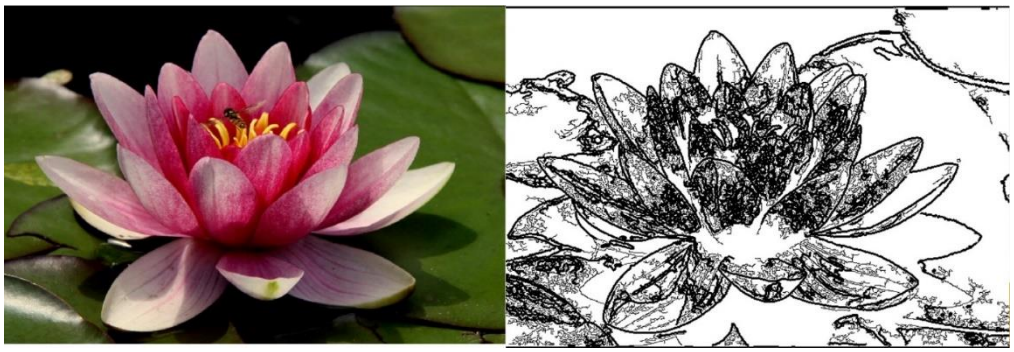


Figure 37 detection example

Cow

1 of 5 stars. A typical example of failure. Smoothing methods were not able to clean the grass and therefore so many unwanted edges were found that the real objects in the image are completely lost. On the other hand, the cow has been detected well enough to be coloured. However, this is not sufficient for the colouring to be marked as successful.

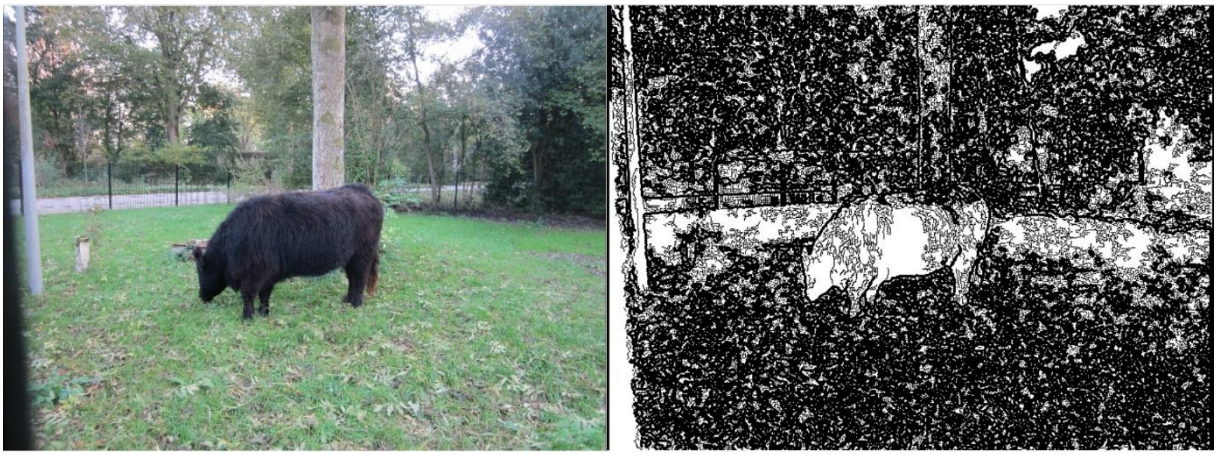


Figure 38 detection example

Preikestolen rock

2.5 of 5 stars. The landscape is detected considerably well. The main issue of this image is the detection of tiny people and their shadows which make the rock look like being composed of noise only.

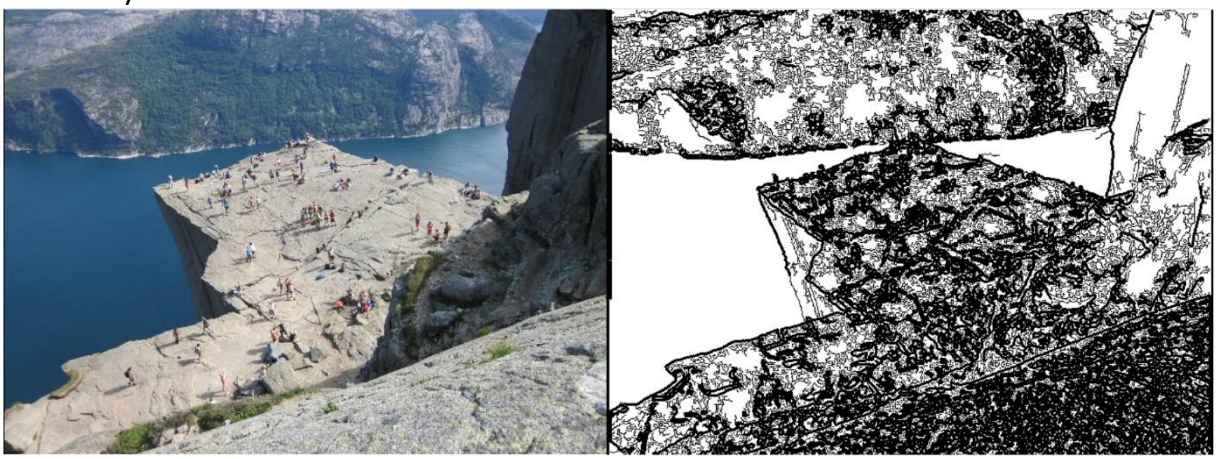


Figure 39 detection example

Flowers

2 of 5 stars. The original image consists of lots of petals and so does the result. However it is hard to divide which petal belongs to which flower. The yellow centres of the flowers are not well detected.

Figure 40 detection example

Comparison of classes

Quality comparison

For each class, a set of 20 representative images had been chosen and their colourings created. Created colourings had been rated with same metric as the examples shown above. Final mark for each class is based on average rating across its set.

Class	Size of set	Median rating	Average rating
Simple Cartoons	21	5	4,714
Advanced Cartoons	20	4,75	4,525
Photos of Cartoons	20	4	3,9
Real Photos	20	3,5	3,75

As been expected, simple cartoon class has the highest average rating. There is a considerably large gap between cartoons and photos. However the rating of real photos is better than being expected.

Time comparison

The detection of colouring is a very complex task and therefore it requires a lot of computation time. In the table below is a comparison of average times for each of classes on testing device [18].

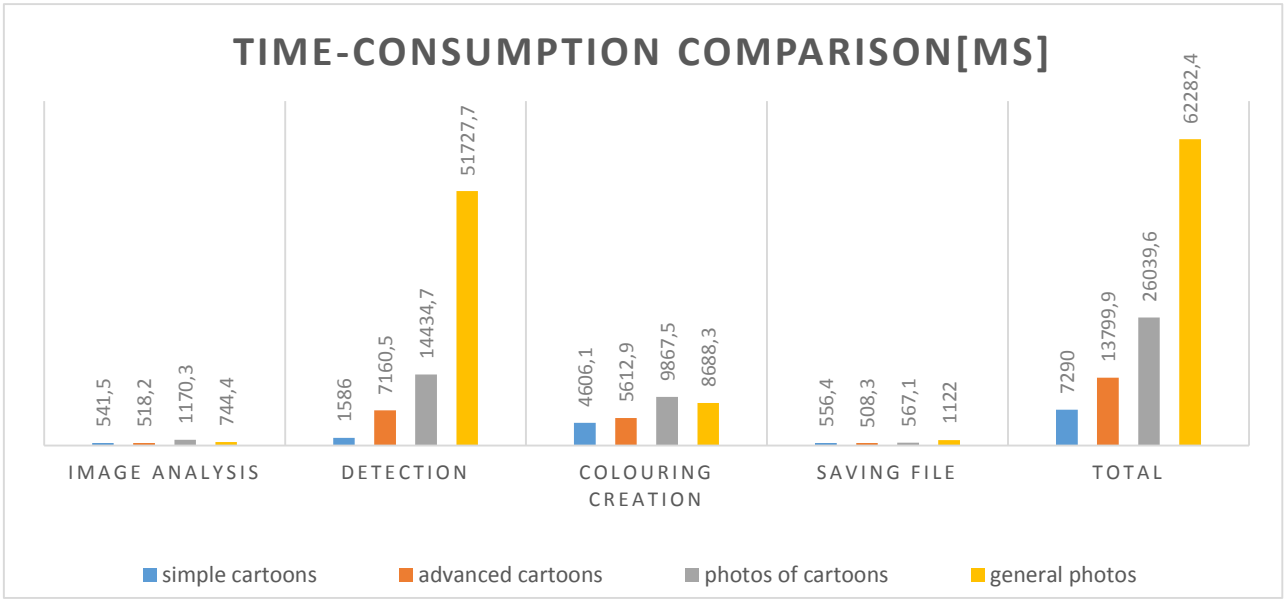


Figure 41 Time-consumption comparison

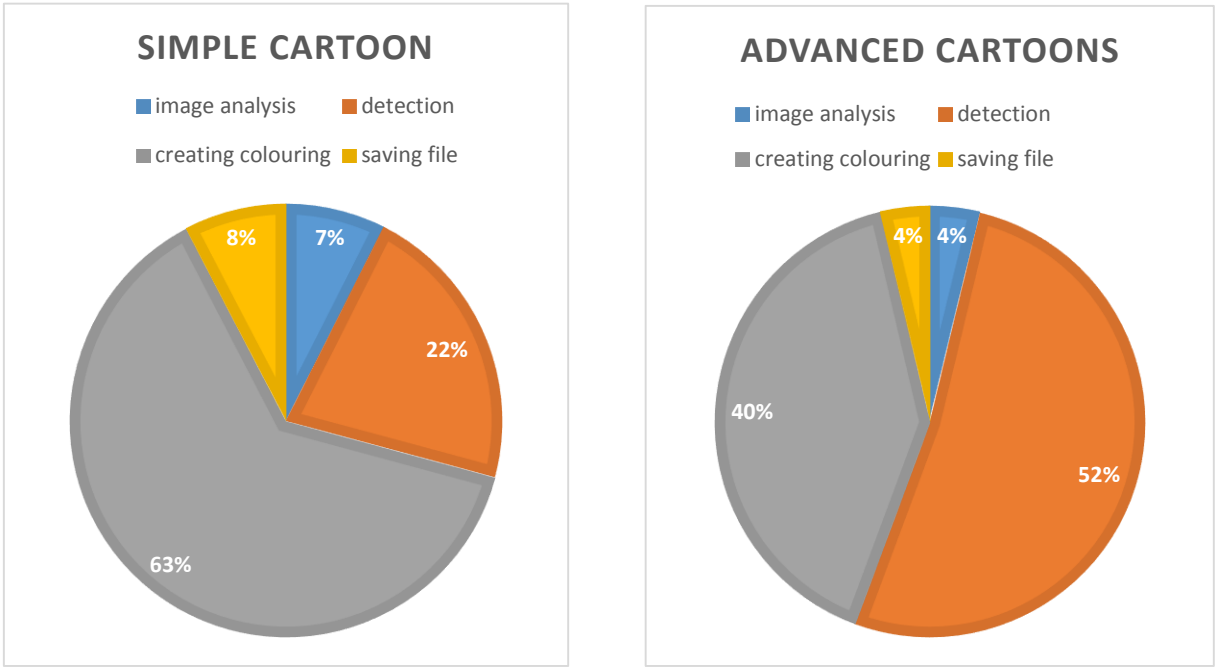


Figure 42 a, b, - Time-consumption of separate classes

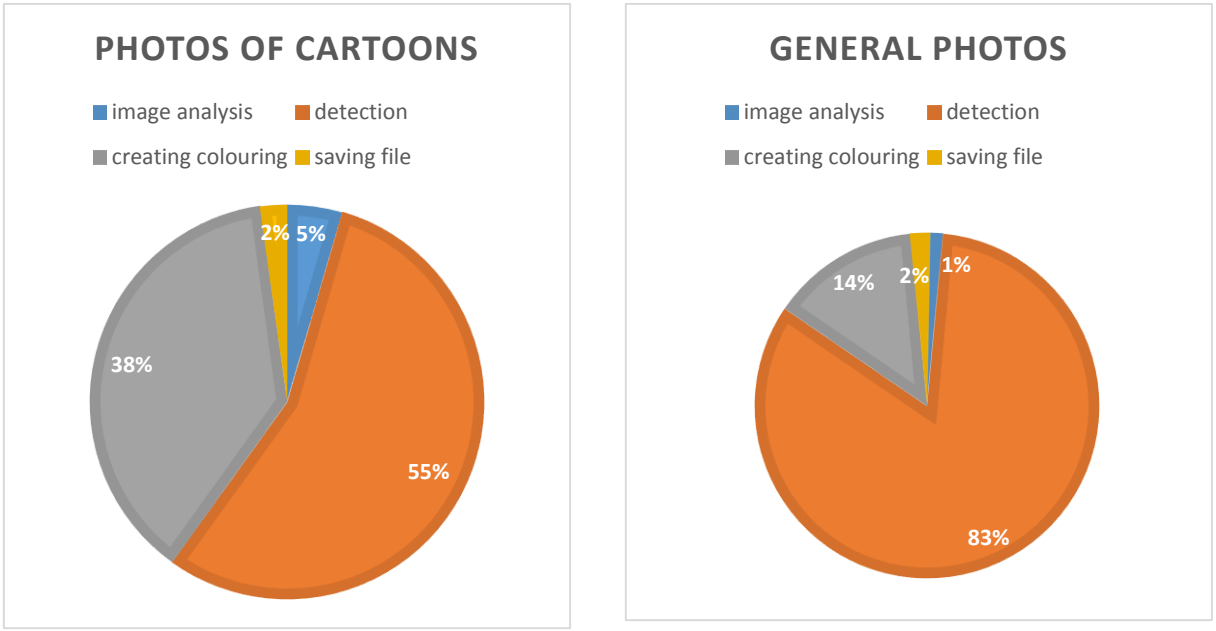
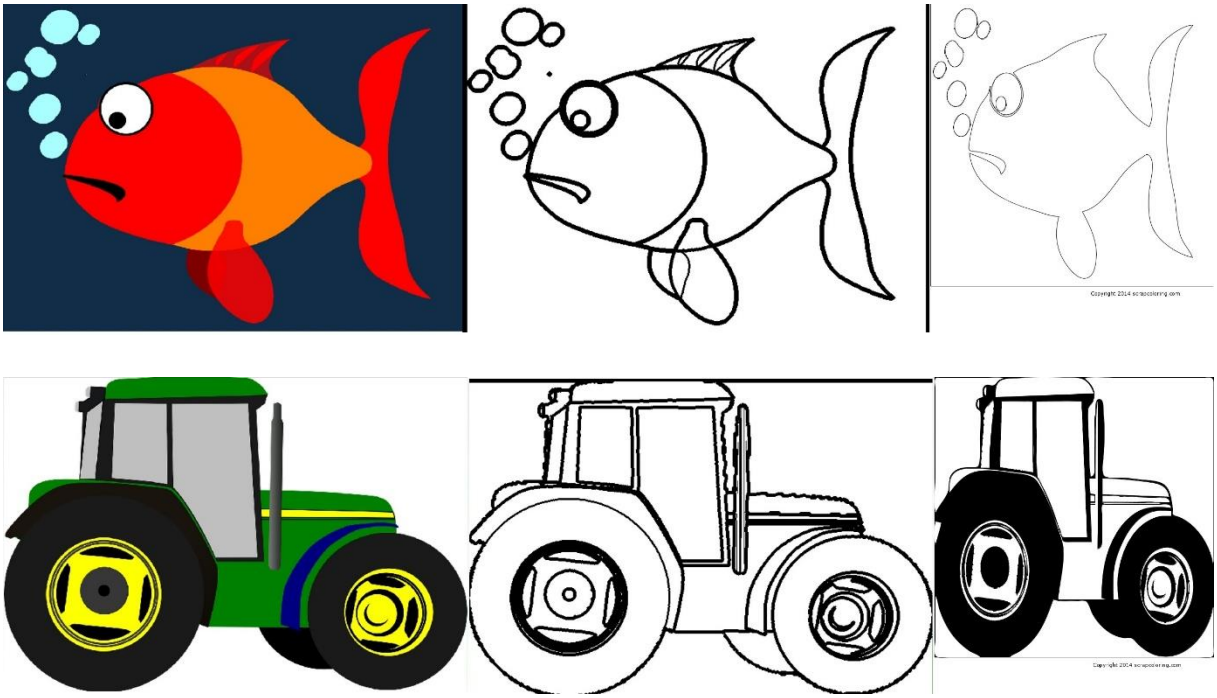
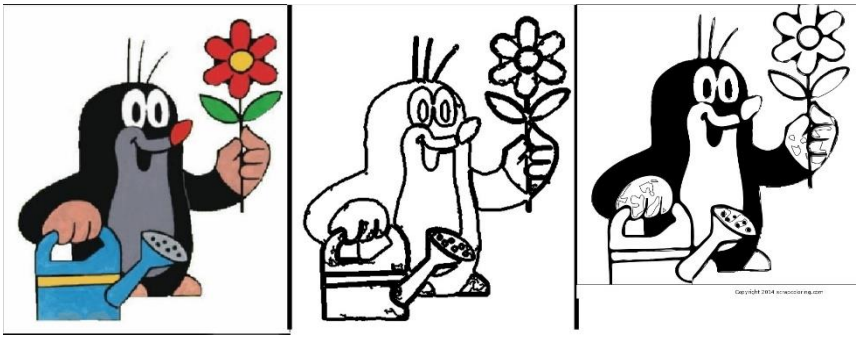


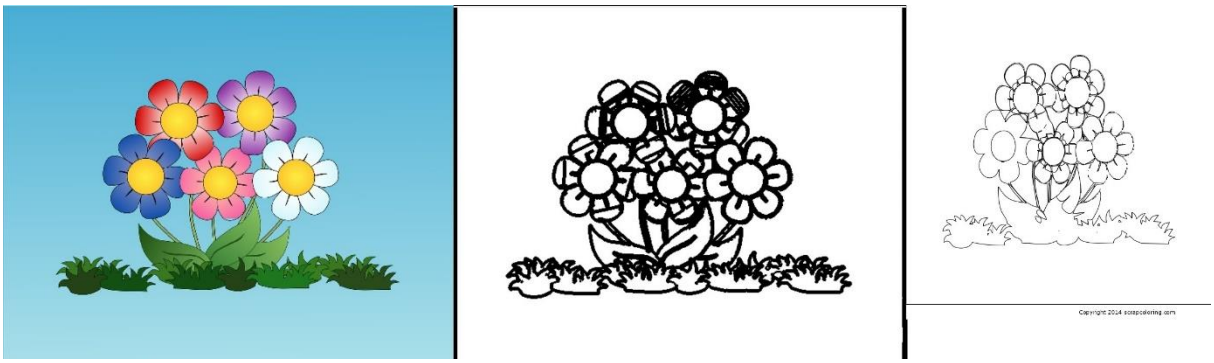
Figure 43 a, b, - Time-consumption of separate classes

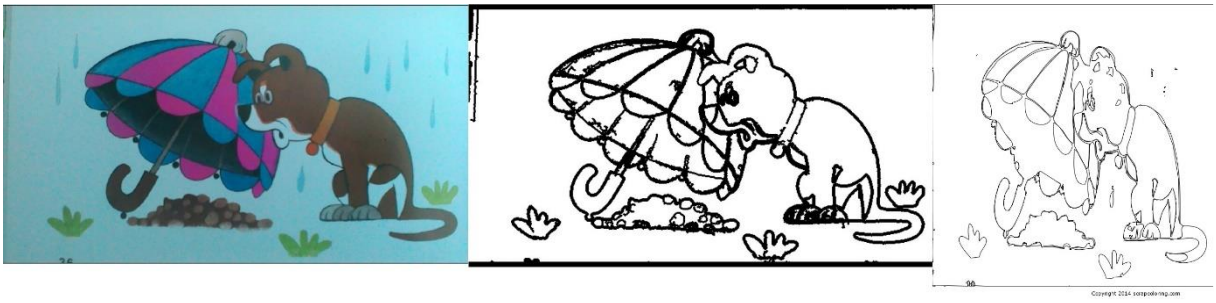
Comparison with Scrap Coloring

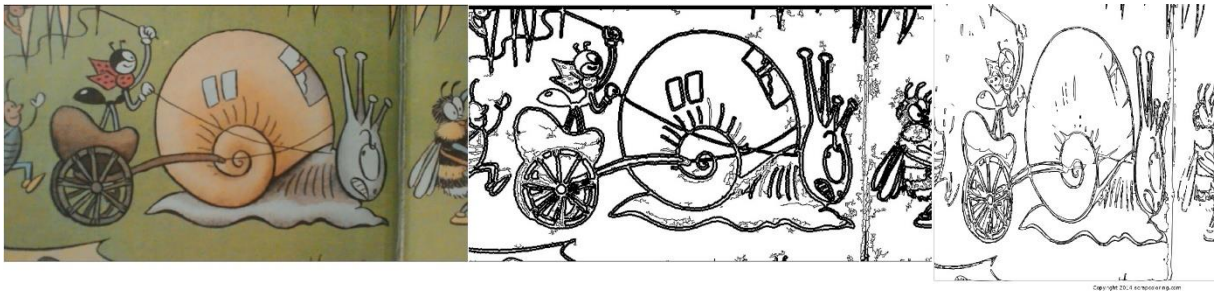
Examples from previous chapter, were used to create colourings at Scrap Coloring as well. And in this chapter the received results are compared. There is no objective metric how to compare colourings and therefore comparison of results will be presented but not commented. Scrap Coloring images were left in their original shape and size in respect to Colour it! results.











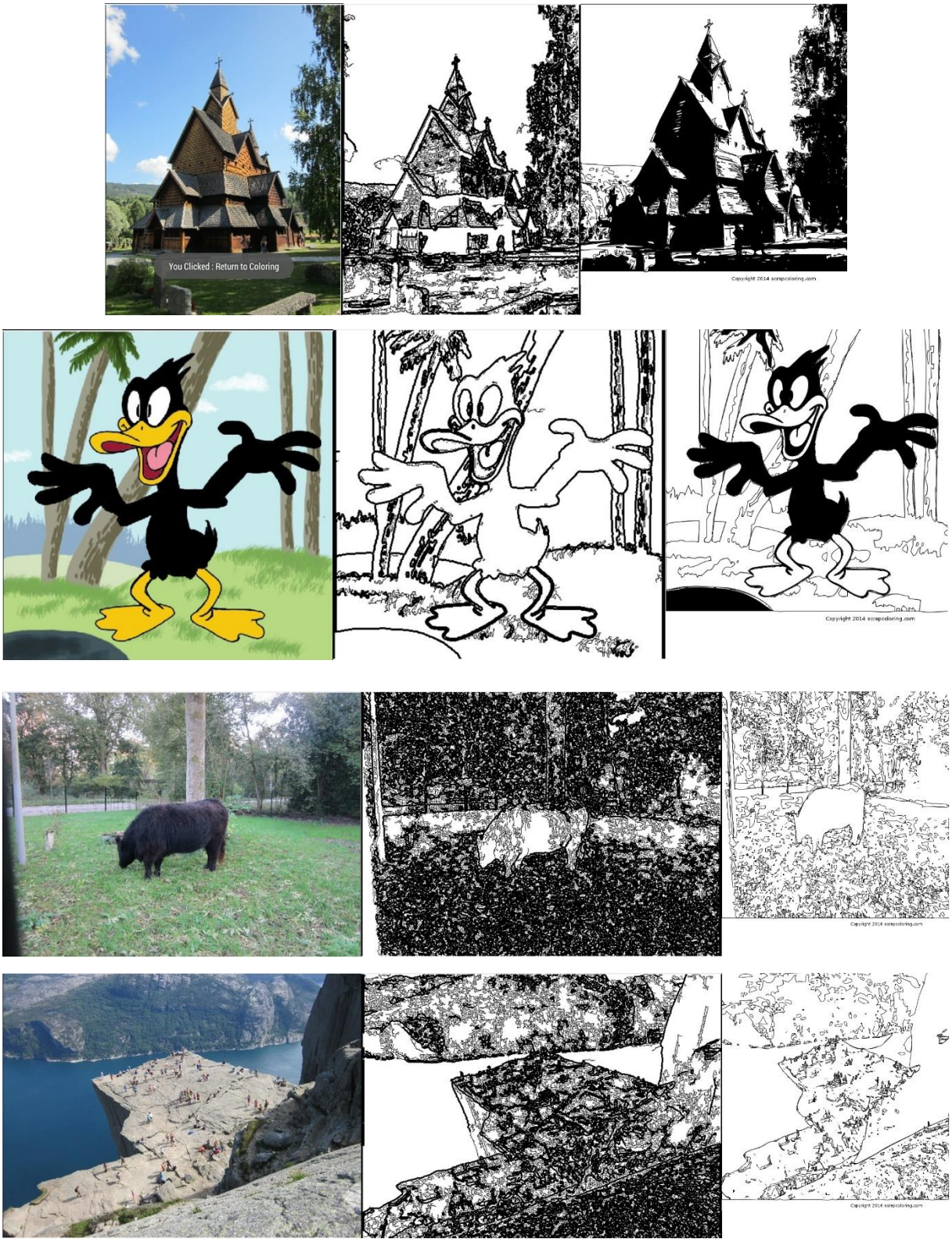


Figure 44 a, b, c, d, e, f, g, h, i, j, k, l – Scrap Coloring and Colour it! comparison

Discussion

This chapter discusses if - and to what extent - the goals were accomplished. The discussion will be presented for each class separately.

This chapter also provides a “cookbook” suggesting which type of image to select when a particular type of result is required.

Result evaluation

Simple cartoons

In most cases the results of simple cartoon pictures are flawless. If the evaluation should be strict then it can be said that some bounds are too thick and thinner ones would be better.

Advanced cartoons

The most common defects consist in the fact that in the results often contains fake lines in the middle of gradient mixtures. This is not a big problem; however for younger users it could be confusing. A bigger issue might be the possible gaps in bounds. In the simple cartoons the edges were without gaps because there were always two solid colours around some continuous part of edge. This does not work for advanced cartoons. The problems are caused by the similarity of object and background colour in some places.

Photos of cartoons

The results are highly dependent on the source quality. It should be mentioned that even low resolution cameras can take good enough photos if the photos are taken with potential usage in mind. In this case it means that the object should be a simple cartoon, otherwise it will be most probably too noisy.

Real photos

Nowadays the quality of average photos is far beyond the necessary level. Therefore in this class the composition of the photo is more important than real density. If the object in the photograph is well distinguished from the background and is not too complicated itself, then the results should be reasonably useful. It is important to mention that this is not the primary source for automatic creation. It is also possible that an experienced user can achieve better results using custom creation.

Creation recommendation

Usually the difficulty of the source image corresponds to the results obtained. However in many cases the image can be simplified during the process. It should be mentioned that significantly simplified images can be gained using custom creation where the user can determine what is truly important for the particular image and his goals.

Creation of a simple colouring

Our results demonstrate that images from the simple cartoon class or photos of simple cartoons should be recommended as a source for simple colourings.

Creation of an advanced colouring

Slightly advanced colourings can be achieved by using advanced cartoons or their photos.

Creation of expert colourings

The most difficult colouring can be gained from real photos. However in many cases the automatic creation will result in failure, and therefore it would be advisable to use custom creation.

Conclusion

Future of project

The results of this application are far above the initial expectations and therefore it is worth further testing and improvement. In near future I would like to upload the application to Google Play and make it freely downloadable.

Possible improvements

During a review of the results I came up with the idea that this application could be used for gathering data for future detection improvement. In future, it would be interesting to implement some statistical recognition algorithms for overall improvement based on gained data. It could be useful to distinguish different classes of images in more accurate ways.

Possible usage

At the beginning the idea was to make an application for children. But when testing GUI I realized that it could be used also by patients during stroke recovery. It could be useful as an alternative and perhaps a less boring way of regaining fine motor skills.

Appendices

Documentation

In this chapter I will discuss the implementation philosophy and provide a more in-depth concept of application than in Introduction chapter. On the other hand the implementation of detection functions used will not be described here. The description of implementation can be found in implementation chapter.

Application life-cycle

In the diagram bellow the grey colour represent actions which are automatically performed. Yellow colour marks a centre of application.

Blue is for managing colouring pages.

Green is for simple tasks which are designed to be handled by children.

Red actions are meant for experienced users only.

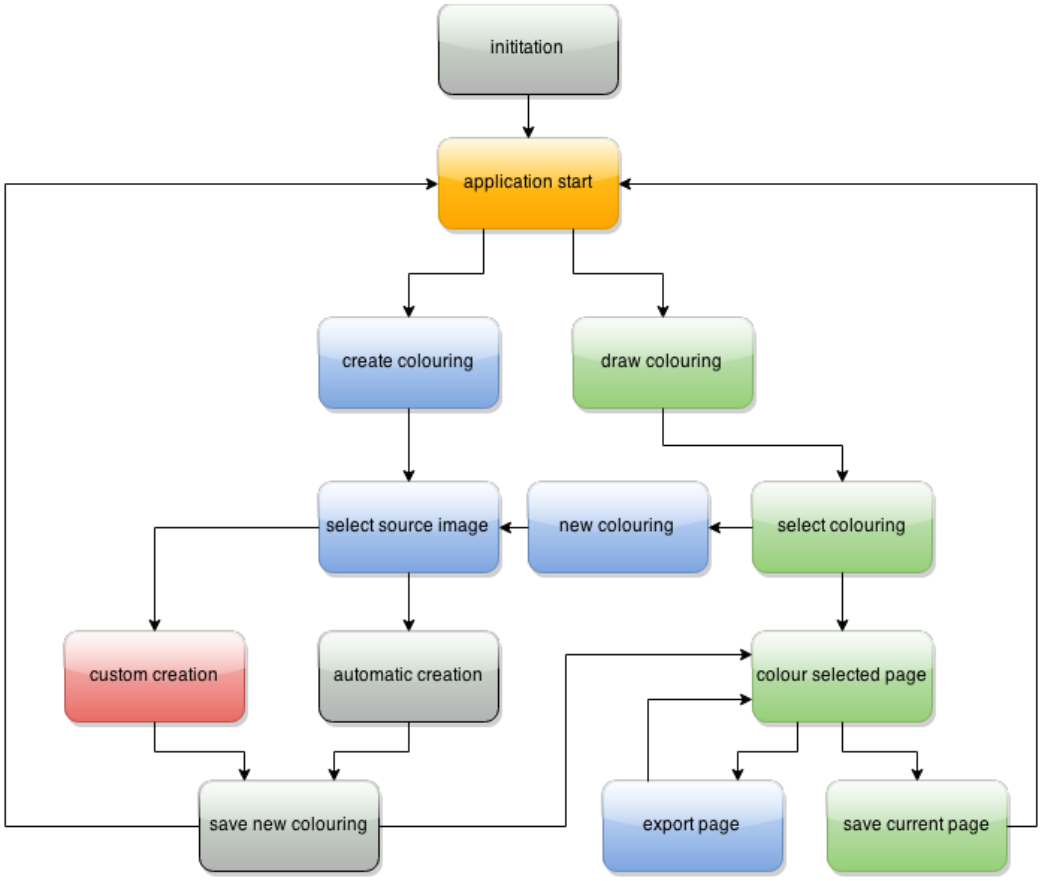


Figure 45 Life-cycle graph

Project overview

The project is divided into several packages for better readability.

Activities package

This package contains all activities used during the application life-cycle. Each activity is responsible for event handling and layout when it is active.

Detection package

This package contains classes which are the only ones that actually use OpenCV library functions. They provide more suitable API or interface for this work and therefore guarantee better readability of code for programmers who are not familiar with OpenCV entities.

Manager package

This package comprises managers which often use tools from detection or utility-tools package to perform the required action. An example of this is saving or loading colouring pages.

Utility-tools package

Here all methods needed are implemented. Methods from this package are divided into separated classes based on their functions for better readability. Classes here are the real hard workers of this application. However they should never been called directly. In this program they are always approached trough managers since methods here have not implemented any exception handling which is not mandatory. This is designed this way for better overall performance. And it also provides better readability.

Since there are only functions which provide results to some inputs without a bound to any object, they are all static.

Entity package

This package contains entities used for this application, namely the entity class Colouring Page which is the fundamental cornerstone of this application.

Colouring Page consists of several datasets which together describe a colouring page. It consist of:

- bitmap of original image, for the purpose of showing a template
- bitmap of detected colouring
- bitmap of current state of colouring
- bitmap of masks with IDs of surfaces to which target pixel belongs
- set of colours which are detected from the original image

Programming philosophy

As a device application, it performs actions based on user inputs. Whenever an event is started and the input is recognized, the active Activity calls one of managers to perform a corresponding action and when manager returns the result the activity will make it appear. Whenever a manager is called, it uses tools from utility-tools or detection package to perform desired actions while being responsible for all exception handling as well. At this level the manager must make sure to use tools properly since tools themselves have no exception handling built in. Those tools are usually short functions like resizing bitmap and so on.

For better understanding I present an example:

Let's say a user is in the middle of colouring a page. Therefore there is some state of colouring page, some selected colour. Let's call this knowledge a State. The user performs an action to colour a new area by touching the screen at the desired location. The active Activity recognizes that input is at the location [X; Y] and that the user wishes to colour the area. Therefore it calls Paint Manager (PM) with providing pointers to current state and location of user input.

PM checks if its input is valid and calls a method which fills the target area and returns a new state. Then PM checks the result and sets a new state as the current and sends a response back to Activity that computation is over. At this moment Activity switches the old colouring for a new one which has the desired area filled. And the application is waiting for the next user input.

User manual – Colour it!

Figure 47 Colour it! icon

Colour it! application allows to create and colour infinite number of new and unique colourings. Due to simple and intuitive graphical layout it is easy to use even for toddlers! Usage is very simple:

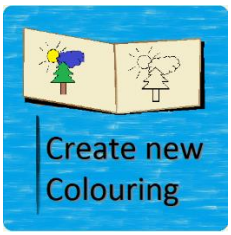


Figure 46 create new

1. Select Create new colouring
2. Select an image which you would like to colour
3. Colour it!

Do you want to continue in colouring of some previously created and unfinished colouring pages? Just simply:

Figure 48 continue colouring

1. Click open Colouring
2. Select the desired colouring
3. Continue where you stopped.

Are you a more experienced user or would you like to know how does it actually work? Then why don't you try it yourself? For curious users there is the advanced creation available. The advanced creation allows to use complicated detection methods in very simple way. Just select a method and see what will happen. You can compare your detection with the automatic process and compete which is better! Usage is again very simple:

Figure 49 advanced creation

1. Select advanced Creation
2. Select an image
3. Apply methods from toolkit until you get a perfect colouring
4. Colour it!

Did something went wrong and you would do something differently? Feel free to reset and start again!

Colouring screen

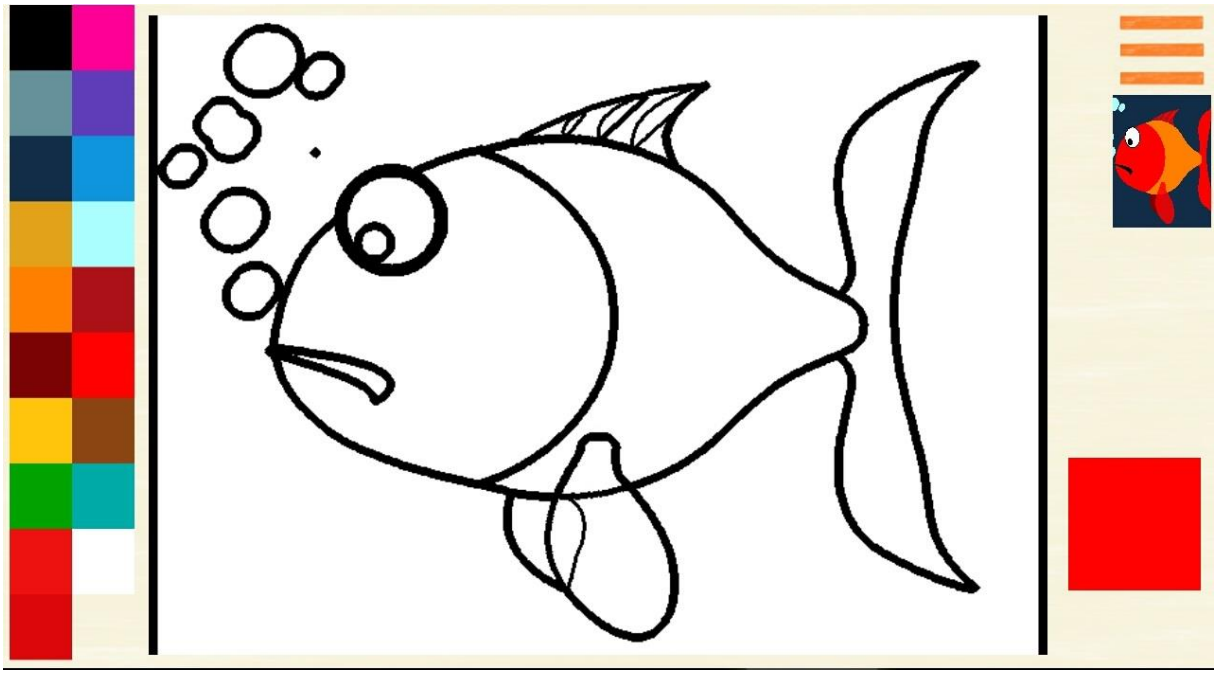


Figure 50 colouring screen example

Colouring environment is very simple to use. On the left side you can see a palette with colours specifically chosen for this colouring. Just click on desired colour and see that square in right bottom corner changed its colour to yours.

In the middle is your colouring. Just click to some area to fill it with selected colour. Yes it is that simple!

Figure 51 colour anything!

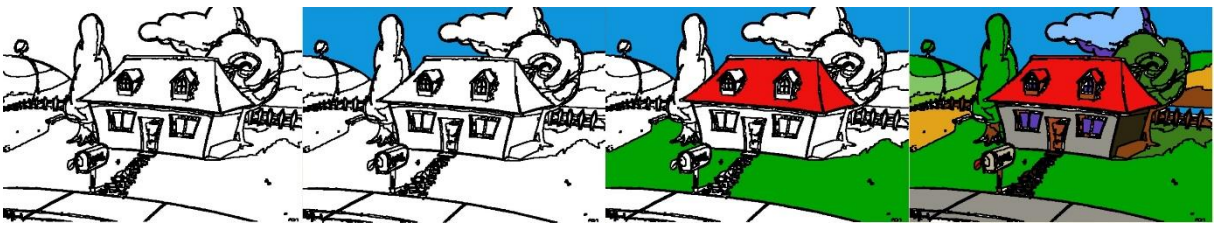
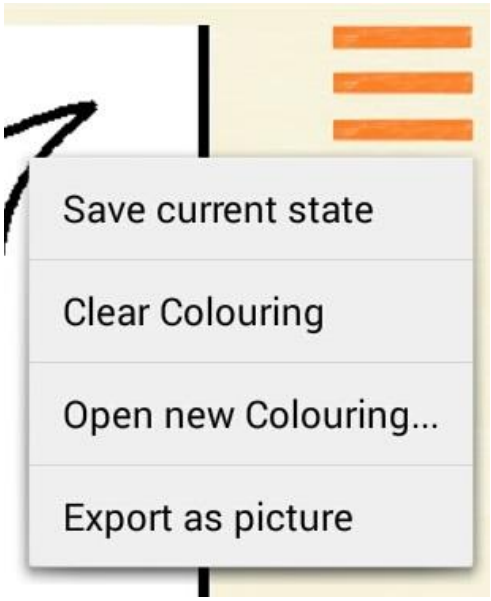


Figure 52 colouring is really simple

Would you like to remind how the original looked like? No problem, just click the miniature of original image. You can find it on the right side in the upper part. Then click the miniature again to return to colouring.



You can find a menu which contains additional features, just above the miniature. From the menu you can:

1. save current progress for later
2. clear the whole colouring to start again
3. open new colouring
4. Or export the image, it will save the colouring as a picture. Then you can print it and colour with pencils or share it with your dearest.

Figure 53 colouring menu

Advanced creation

Advanced creation screen allows you to unleash full potential of detection methods to enhance the detection of more sophisticated images. There is a powerful toolbox prepared, containing variety of methods and packed into simple control panel.

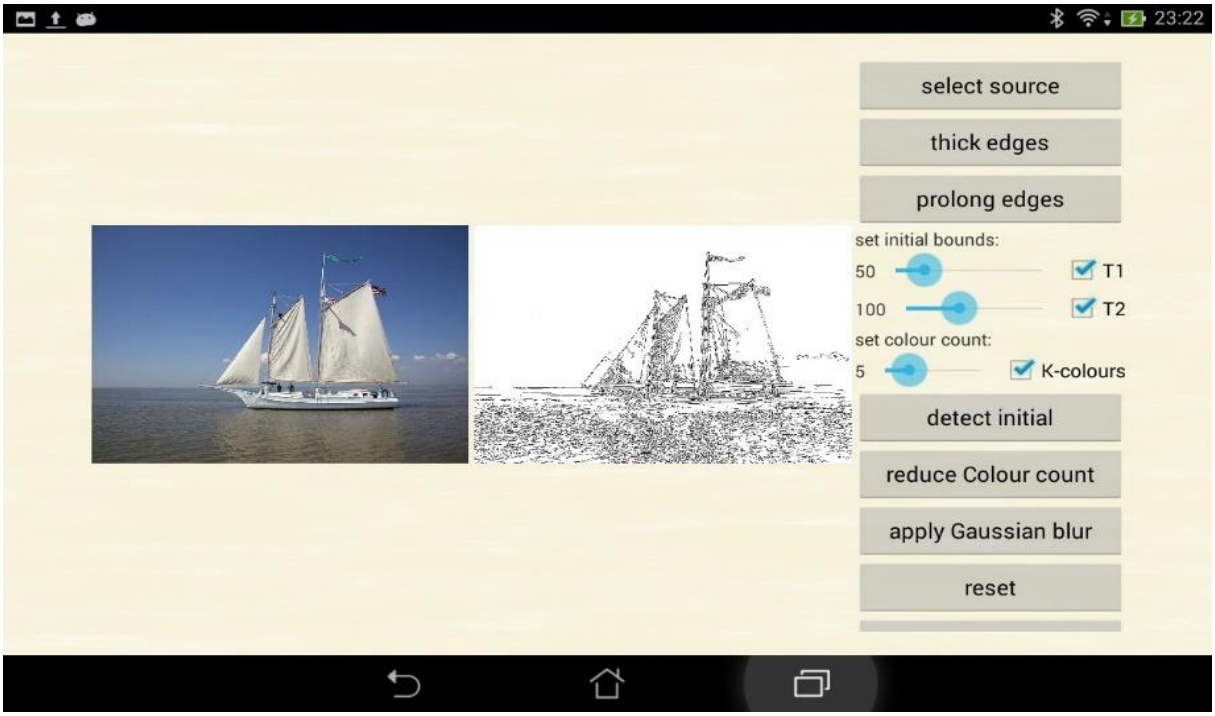


Figure 54 advanced creation example

Toolbox consists of following methods:

1. Reduce colour count – It reduces number of colours present in the image to selected number
2. Apply Gaussian blur - Is your image noisy? Then it is time to use this. It will help to the rest of tools in their performance.
3. Detect initial – use sliders to properly select what is important and what is not
4. Thick edges – detected edges contains small gaps? Then you might want to have the detected edges a bit thicker. Feel free to apply it multiple times.
5. Prolong edges - detection is still missing some key points? By prolonging the edges you might find some missing contours.
6. Reset – did something went wrong? Feel free to try it again with different setup.
7. Save colouring – is your detection flawless? Save it, so you or your children can colour it any time later.

Installation

The installation of this application is done by an .apk file which after transferring to device automatically installs the application. However since this application uses OpenCV which is distributed as a self-standing application, it is necessary to download the OpenCV library as well. The easiest way to do this is to connect the device to Google Play and then start the Colour it! application. If there is no library installed it will pop a window which will ask for installing it. If you allow this it will be downloaded and installed without any other actions. However internet connection will be required.

Resources

Testing images were collected across Pixabay.com web which provides pictures without reuse restrictions.

[1] Šonka M., Hlaváč V., Boyle R.: Image Processing, Analysis and Machine vision, 3rd edition, Thomson Learning, Toronto, Canada, 2007, ISBN

[2] Robert Laganiére: OpenCV 2 Computer Vision, Application Programming Cookbook, 1st edition, Packt Publishing Ltd., Birmingham, UK, 2011, ISBN 978-1-849513-24-1

[3] Pavel Herout: Učebnice jazyka Java, 5th extended edition, Kopp publishing, České Budějovice, Czech Republic, 2013, ISBN 978-80-7232-398-2

[4] Nudelman G.: Android Design Patterns, 1st edition, Wiley, Indianapolis, USA, 2013

[5] L. Lucchese, S.K. Mitra, Color Image Segmentation: A State-of-the-Art Survey, [online], 2007, [accessed 15.5.2015] available at:

<http://ultra.sdk.free.fr/docs/Image-Processing/filters/Color%20Image%20Segmentation-A%20State-of-the-Art%20Survey.pdf>

[6] Itseez team, OpenCV library documentation, [online] 2015, [accessed 10.5.2015] available at: < http://docs.opencv.org/java/>

[7] Google Inc.: Android developer, [online] 2015, [accessed 14.5.2015] available at:

< http://developer.android.com/training/index.html>

[8] David Arthur, Sergei Vassilvitskii: k-means++: The Advantages of Careful Seeding, [online] 2007, [accessed 5.3.2015] available at:

<http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

[9] A. Greensted, Otsu Thresholding, [online] 2010, [accessed 13.5.2015] available at:

< http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html>

[10] Braxmeier, Steinberger GbR: Pixabay, [online] 2015, [accessed 15.5.2015] available at:

<http://pixabay.com/>

[11]Elsa and Stéphane Gigandet: Scrap Coloring, [online] 2010, [accessed 8.5.2015] available at:

<http://scrapcoloring.com/>

[12]Google Inc.: Google Play [online] 2015, [accessed 14.5.2015] available at:

<https://play.google.com/store>

[13]O. Sekora: Knížka Ferdy mravence, 6th edition, Albatros, Prague, Czech Republic, 1962, ISBN 13-864-78

[14]Z. Miler, H. Doskočilová: Krtek a paraplíčko,4th edition, Albatros, Prague, Czech Republic, 2002, ISBN 13-712-002

[15] Arthur, D. and Vassilvitskii, S. (2006), "How slow is the k-means method? “ ACM New York, NY, USA: 144–153

[16] Digital trends Inc.: Android claims 81.5% of the global smartphone OS market in 2014, iOS dips to 14.8%, [online] 2015, [accessed 15.5.2015] <http://www.digitaltrends.com/mobile/worldwide-domination-android-and-ios-claim-96-of-the-smartphone-os-market-in-2014/>

[17]Wikipedia contributors, Wikipedia, The Free Encyclopaedia: Determining the number of clusters in a data set, [online] 5.5.2015, [accessed 18.5.2015]

<http://en.wikipedia.org/w/index.php?title=Determining_the_number_of_clusters_in_a_data_set&oldid=661006527 >

[18] Testing device: ASUS MeMO Pad 8 ME181CX 16GB:

Tablet - Intel Atom Quad Core Z3745 1.86GHz Bay Trail, 8" 1280x800 IPS, 1GB RAM, internal storage 16GB, 2x camera 2MPx + 0.3MPx, Google Android 4.4

<http://www.asus.com/us/Tablets/ASUS_MeMO_Pad_8_ME181C/>