VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

MOBILNÍ APLIKACE PRO NASKENOVÁNÍ HRY HITORI Z NOVIN A JEJÍ DOHRÁNÍ

MOBILE APPLICATION FOR SCANNING HITORI FROM NEWSPAPERS AND FINISHING IT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MONIKA BUREŠOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ DYK**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Studentka: **Burešová Monika**
Program: Informační technologie
Název: **Mobilní aplikace pro naskenování hry Hitori z novin a její dohrání**
Mobile Application for Scanning Hitori from Newspapers and Finishing It
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy zpracování obrazu. Zaměřte se zejména na problematiku detekce a rozpoznání tištěných číslic.
2. Prostudujte problematiku návrhu a tvorby mobilních aplikací pro android.
3. Navrhněte mobilní aplikaci, která nasnímá vytištěnou křížovku Hitori a navrhne kroky pro její vyřešení.
4. Implementujte minimalistickou použitelnou verzi řešené aplikace.
5. Otestujte aplikaci na testovacím vzorku uživatelů.
6. Analyzujte zpětnou vazbu uživatelů a realizujte iterativní úpravy řešené aplikace.
7. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání,
- rozpracovaný bod 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Dyk Tomáš, Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 28. ledna 2021

Abstrakt

Tato bakalářská práce se zabývá vývojem mobilní aplikace s prvky počítačového vidění a umělé inteligence, která je určena pro operační systém Android. Aplikace slouží pro naskenování hry Hitori z fotografie a její následné dohrání na mobilním zařízení. Navíc tato aplikace poskytuje nápovědu, navrácení změny nebo znázornění porušení pravidel hry. Pro zpracování obrazu je použita knihovna OpenCV, pro samotné rozpoznání číslic pak knihovna Tesseract. Naskenované hry jsou ukládány do databáze, která je implementována pomocí knihovny Room. Správné řešení Hitori je hledáno kombinací algoritmu backtracking (zpětného vyhledávání) a známých technik pro řešení této hry. Výsledná aplikace je dostupná na Google Play pod názvem Hitori Scan & Play.

Abstract

This bachelor's thesis deals with development of mobile application with elements of computer vision and artificial intelligence, which is designed for Android operating system. The application is used for scanning Hitori from photo and then finishing it on a mobile device. Moreover, this application provides hint, undo or shows a violation of rules of the game. The OpenCV library is used for image processing, Tesseract library is used for digits recognition. Scanned games are stored in a database, which is implemented using Room library. Hitori solution is found by combination of backtracking algorithm and known techniques for solving this game. The final application is available on Google Play and is named Hitori Scan & Play.

Klíčová slova

hitori, logická hra, počítačové vidění, umělá inteligence, mobilní aplikace, rozpoznávání čísel, zpracování obrazu, Android

Keywords

hitori, logic puzzle, computer vision, artificial intelligence, mobile application, numbers recognition, image processing, Android

Citace

BUREŠOVÁ, Monika. *Mobilní aplikace pro naskenování hry Hitori z novin a její dohrání.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Dyk

Mobilní aplikace pro naskenování hry Hitori z novin a její dohrání

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Tomáše Dyka. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....
Monika Burešová
11. května 2021

Poděkování

Chtěla bych poděkovat vedoucímu mé bakalářské práce Ing. Tomáši Dykovi za odborný přístup a cenné rady. Také bych ráda poděkovala všem, jež byli ochotni podílet se na uživatelském testování aplikace, která je produktem této práce.

Obsah

Kapitola 1

Úvod

Cílem této bakalářské práce je vytvořit aplikaci pro platformu Android umožňující naskenování hry Hitori z fotografie a její dohrání na mobilním zařízení. To lze rozdělit do dvou menších logických celků, kterými jsou rozpoznání obrazu a samotná hra.

Zpracování obrazu je poměrně výpočetně náročné, avšak v dnešní době je potřebný výpočetní výkon docela dostupný. Díky tomu se tento obor těší velké popularitě a rychlému rozvoji. S pojmem zpracování obrazu je úzce svázáno počítačové vidění, které se snaží napodobit lidské vidění. Počítačové vidění je využíváno třeba u autonomních vozidel. To se může čtenáři zdát jako vzdálená budoucnost, ale například parkovací asistenti jsou dnes už součástí základní výbavy.

Aplikace by měla uživateli nabídnout několik výhod oproti řešení Hitori na papíře. Jednou z nich je, že uživatel nepotřebuje žádné psací potřeby. Další výhodou je možnost opravy, takže uživatel neskončí s doškrtaným kusem papíru, ze kterého už ani nejde nic přečíst. Ale hlavně nabídne nápovědu, což se může hodit u náročnějších zadání. Pro vytvoření nápovědy je potřeba využít algoritmy napodobující lidské myšlení, aby uživatel po použití nápovědy mohl na další krok už přijít sám.

Kapitola 2 je věnována samotné hře Hitori, jejím pravidlům a způsobům řešení. Kapitola 3 pojednává o základních praktikách zpracování obrazu a knihovnách, které toto umožňují. Kapitola 4 je věnována základům umělé inteligence a nabízí možnosti, jak hledat správné řešení hry Hitori. Kapitola 5 popisuje operační systém Android. V kapitole 6 jsou zhodnoceny vybrané aplikace, které řeší podobnou problematiku jako výsledná aplikace. Kapitola 7 předvádí návrh aplikace. Kapitola 8 popisuje implementaci mého řešení. Kapitola 9 popisuje průběh a výsledky testování aplikace.

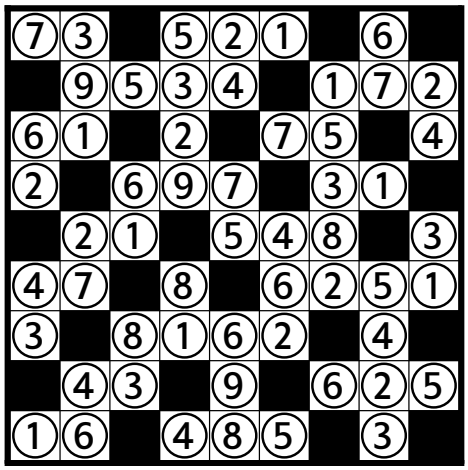
Kapitola 2

Hitori

Hitori je logická hra s čísly. Hrací plochu tvoří tabulka rozdělená na 9×9 polí, která jsou dopředu vyplněna čísly. Existují i varianty lišící se velikostí hrací plochy. Hitori vzniklo v japonské společnosti Nikoli, která se specializuje na logické hry.

Na obrázku 2.1 je příklad možného zadání Hitori a jeho řešení. Cílem hry je smazat **(vybarvit)** čísla tak, aby se číslo vyskytovalo v každém řádku nebo sloupci nejvýše jednou, pokud vůbec, a aby zůstávající čísla byla vzájemně propojena. Pro usnadnění řešení se používá technika označování polí, která nesmí být vybarvena. V případě řešení hry v papírové podobě se proto používá **kroužkování**.

7	3	2	5	2	1	1	6	7
9	9	5	3	4	1	1	7	2
6	1	6	2	2	7	5	6	4
2	2	6	9	7	6	3	1	2
7	2	1	9	5	4	8	6	3
4	7	6	8	6	6	2	5	1
3	7	8	1	6	2	8	4	6
2	4	3	9	9	1	6	2	5
1	6	8	4	8	5	8	3	6



Obrázek 2.1: Ukázka zadání hry Hitori a jejího řešení[1]

Pro úspěšné dokončení hry musí být splněna tři následující pravidla:

1. V každém sloupci či řádku se každé číslo smí vyskytovat nejvýše jednou.
2. Smazaná pole se nesmí dotýkat vodorovně, ani svisle (mohou se dotýkat přes roh).
3. Čísla, která mají zůstat, musí být všechna propojena (tvoří jednolitou plochu).

[1] Obrázky převzaty z časopisu *Mensa* [24]

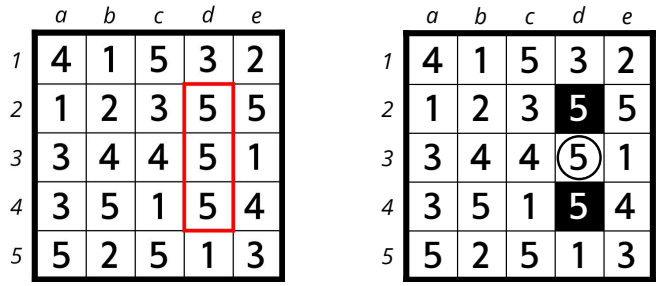
2.1 Techniky řešení

Existuje několik technik pro vyřešení Hitori. V této části jsou uvedeny ty nejzákladnější, které fungují jako odrazový můstek. Pro vyřešení složitějších zadání však nemusí být postačující.

2.1.1 Počáteční techniky

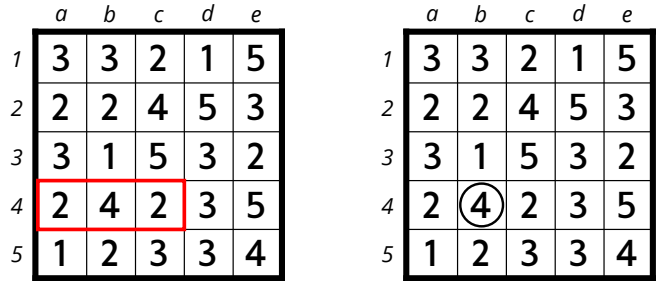
Níže je uvedeno několik technik, se kterými je vhodné začít. Kromě toho se doporučuje si zakroužkovat čísla, která se v řádku nebo sloupci vyskytují pouze jednou. Po provedení těchto technik je vhodné přejít na základní techniky.

1. **Hledání trojic**
   Ve sloupci *d* se vyskytuje trojice pětek. Jediná možnost jak zajistit, aby se neopakovaly je smazání krajních pětek a ponechání prostřední.



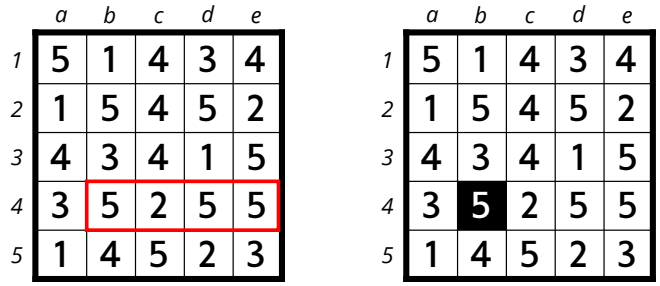
Obrázek 2.2: Hledání trojic

2. **Hledání pole mezi dvěma stejnými čísly**
   V řadě 4 je dvojice dvojek, které odděluje pouze jedna buňka. Je jisté, že jedna z dvojek musí být smazána. Ikdyž ještě není známo, která z nich to bude, je zřejmé, že buňka nacházející se mezi nimi musí být ponechána. Jinak by bylo porušeno pravidlo číslo 2.



Obrázek 2.3: Hledání pole mezi dvěma stejnými čísly

3. **Indukce párů**
   Pokud jsou vedle sebe dvě stejná čísla, která mají v řádku nebo sloupci další výskyt, je potřeba smazat všechny další výskyty tohoto čísla. Které číslo z dvojice má být smazáno zatím není známo.

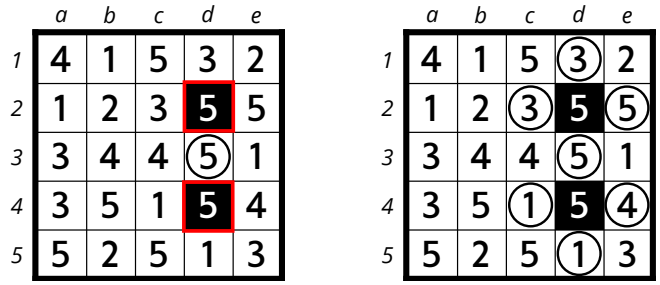


Obrázek 2.4: Indukce párů

2.1.2 Základní techniky

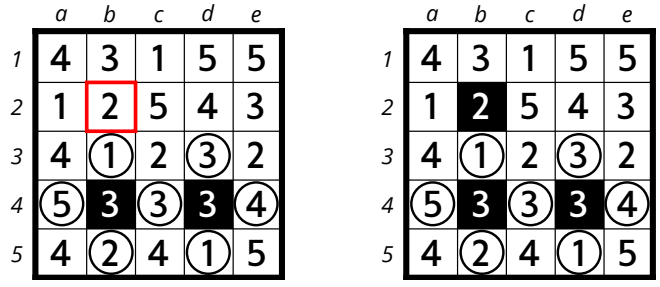
Po označení některých polí pomocí počátečních technik je vhodné postupovat podle následujících technik tak, aby nebylo porušeno žádné ze tří pravidel.

1. **Kroužkování kolem smazaných polí**
   Druhé pravidlo Hitori říká, že smazaná pole se nesmí dotýkat. Z toho vyplývá, že pole kolem smazaného pole musí být zakroužkována.



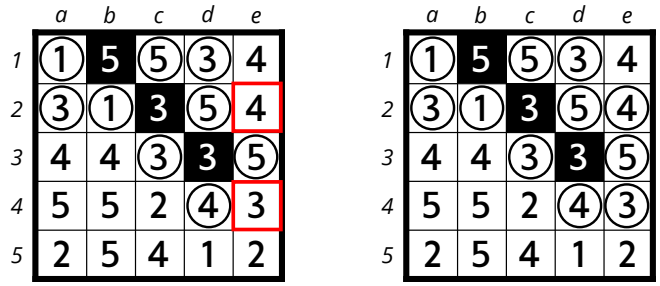
Obrázek 2.5: Kroužkování kolem smazaných polí

2. **Mazání**
   V tomto případě je v poli *b5* zakroužkované číslo 2. To znamená, že se ve sloupci *b* nesmí vyskytovat znovu. Proto je nutné další výskyt čísla 2 smazat.



Obrázek 2.6: Mazání

3. **Kroužkování kvůli zabránění odříznutí**
   Třetí pravidlo hitori říká, že zakroužkovaná pole musí tvořit jednolitou plochu. Tomu lze zabránit zakroužkováním polí, která vytvoří cestu.

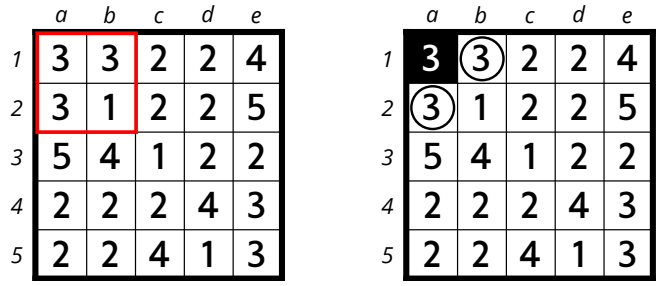


Obrázek 2.7: Kroužkování kvůli zabránění odříznutí

2.1.3 Rohové techniky

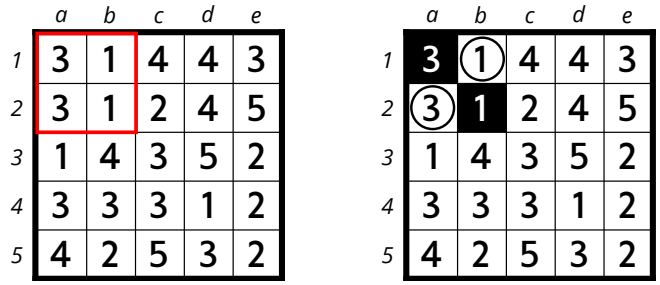
V zadáních Hitori bývají často situace, kdy je v rozích potřeba využít některé z pokročilejších technik. Proto je vhodné zkontrolovat všechny čtyři rohy, jestli v nich nedochází k některé z následujících situací.

1. **Rohová technika 1**
   Pokud by byla smazána pole *a2* a *b1*, zůstalo by pole *a1* osamoceno, což porušuje třetí pravidlo. Tudíž musí být smazáno pole *a1*.



Obrázek 2.8: Rohová technika 1

2. **Rohová technika 2**
   Je třeba zabránit odříznutí, stejně jako u rohové techniky 1. Stejně by vypadalo řešení i v případě, kdy by byla všechna čtyři čísla v rohu stejná.



Obrázek 2.9: Rohová technika 2

Další pokročilé techniky lze najít v [3], odkud byly převzaty i obrázky.

Kapitola 3

Zpracování obrazu

V této kapitole je podán přehled některých základních metod zpracování obrazu, které jsou využity například pro detekci hracího pole Hitori. Jsou zde také popsány knihovny, které toto umožňují.

3.1 Adaptivní prahování

Vstupem je obrázek ve stupních šedé. Pro převod barevného obrázku do stupňů šedi je potřeba určit intenzitu každého pixelu. Výsledná hodnota závisí na všech třech barevných složkách (R: červená, G: zelená, B: modrá). Pro převod z barevného modelu RGB na stupně šedi platí:

$$I_v = 0,299R + 0,587G + 0,114B$$

Protože lidské oko je citlivější na zelenou barvu, má tato barva větší váhu.

Výstupem adaptivního prahování je segmentovaný obrázek:

$$g(x,y) = \begin{cases} 1 & \text{pro } f(x,y) > T \\ 0 & \text{pro } f(x,y) \leq T. \end{cases}$$

kde:

- $f(x,y)$ je funkce prahování
- T je hodnota prahu,
- x a y jsou souřadnice zpracovávaného pixelu
- 1 představuje bílou barvu, 0 černou

Vstupní obrázek ve stupních šedé je převeden do histogramu určujícího intenzitu jednotlivých pixelů. Pixely, které mají v histogramu menší hodnotu než práh, představují pozadí, a ty které mají vyšší hodnotu, zase popředí. Výstupem je černobílý obrázek (binární).

Při adaptivním prahování je obrázek navíc rozdělen do několika částí (obdélníků), ve kterých se prahování provádí [15].

3.2 Cannyho hranový detektor

Tento detektor vytvořil John F. Canny. V roce 1986 jej popsal v publikaci *A Computational Approach to Edge Detection* [14]. Jeho výstupem je obrázek s černým pozadím a bíle vykreslenými hranami.

Existují tři kritéria, která mají vliv na výkon detektoru. Prvním z nich je minimální počet chyb. Je důležité, aby hrany, které se nacházejí v obrázku nebyly vynechány a aby nebyly detekována hrany, které hranami nejsou. Druhé kritérium říká, že poloha hrany musí být určena co nejpřesněji. Nakonec poslední kritérium je, že každá hrana smí být detekována pouze jednou (nesmí docházet ke zdvojení).

Algoritmus detektoru lze rozdělit do čtyř kroků, které jsou podrobněji popsány níže:

1. eliminace šumu,
2. určení gradientu,
3. nalezení lokálních maxim,
4. eliminace nevýznamných hran.

3.2.1 Eliminace šumu

Pro eliminaci šumu se nejčastěji používá Gaussův filtr, který využívá konvoluci s maskou, která se skládá z elementů určených Gaussovou funkcí, která má ve dvou dimenzích tvar:

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

kde:

- x a y jsou vzdálenosti od zpracovávaného pixelu
- σ je směrodatná odchylka Gaussova rozdělení
- $g(x, y, \sigma)$ je velikost konvolučního jádra

Gaussova funkce odstraňuje vliv bodů, které mají vzdálenost větší než 3σ od středu. Výsledkem je rozostřený obraz. To pomáhá splnit první kritérium. [22]

3.2.2 Určení gradientu

Určení gradientu znamená určení jeho velikosti a směru. K tomu lze využít Sobelův filtr, který využívá konvoluční masku o velikosti 3x3.

Velikost gradientu:

$$G = \sqrt{E_H^2 + E_V^2}$$

Směr gradientu:

$$\Theta = arctg\left(\frac{E_V}{E_H}\right)$$

E_V a E_H jsou vertikální a horizontální aproximace derivace:

$$E_V = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I$$

$$E_H = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$

I je zpracovávaný obrázek [16].

3.2.3 Nalezení lokálních maxim

Využívá gradienty nalezené v předchozím kroku a vybírá z nich jen ty, které dosahují lokálního maxima. Ty, které lokálního maxima nedosahují jsou odstraněny, aby byly detekovány hrany v místě největšího gradientu. Tím se zajistí splnění druhého kritéria (poloha hrany musí být určena co nejpřesněji).

3.2.4 Eliminace nevýznamných hran

Provádí se pomocí dvojitého prahování – oproti normálnímu prahování 3.1 jsou zde místo jednoho prahu dva: minimální práh T_{min} a maximální práh T_{max}. Pokud je hodnota hrany větší než T_{max}, tak je hrana uznána a označena jako silná. Pokud je menší než T_{min} tak uznána není. Pokud leží v intervalu mezi T_{min} a T_{max}, tak je uznána jen když je v jejím okolí už uznaná hrana.

3.3 Houghova transformace

Tato metoda byla určena pro detekci přímek v obraze, následně byla rozšířena pro nalezení libovolných tvarů, nejčastěji kruhů a elips. V této části je popsán princip této metody pro hledání přímek.

3.3.1 Definice přímky

Přímka je v kartézské soustavě souřadnic popsána pomocí parametrů m (směrnice) a b (úsek) rovnicí:

$$y = mx + b$$

Problémem je, že touto parametrickou rovnicí nelze popsat vertikální přímky (parametr m je v tomto případě nekonečný), proto není toto vyjádření přímky vhodné.

Houghova transformace proto využívá polární tvar parametrického vyjádření přímky:

$$\rho = x \cos\theta + y \sin\theta$$

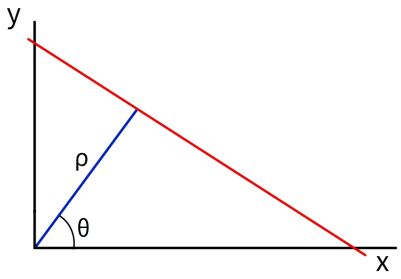
V základním tvaru tedy:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right) x + \left(\frac{\rho}{\sin\theta}\right)$$

kde:

- ρ je vzdálenost od středu souřadnicového systému,
- θ je úhel mezi osou x a přímkou vzdálenosti (znázorněno na obrázku 3.1).

Tato definice přímky umožňuje vyjádřit i vertikální přímky.

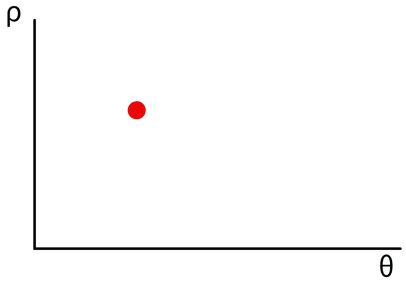


Obrázek 3.1: Přímka (s parametry θ a ρ) v obrazovém prostoru

3.3.2 Houghův prostor

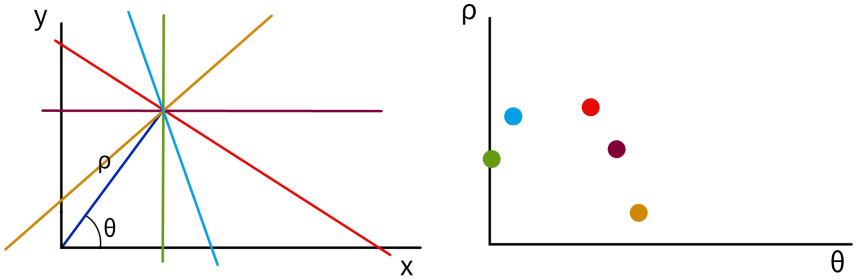
Houghův prostor, také zvaný *akumulátor*, je dvourozměrný graf, kde jedna osa nese vzdálenost ρ a druhá osa úhel θ.

V Houghově prostoru je přímka vykreslena jako jeden bod, viz obrázek 3.2.



Obrázek 3.2: Přímka v Houghově prostoru

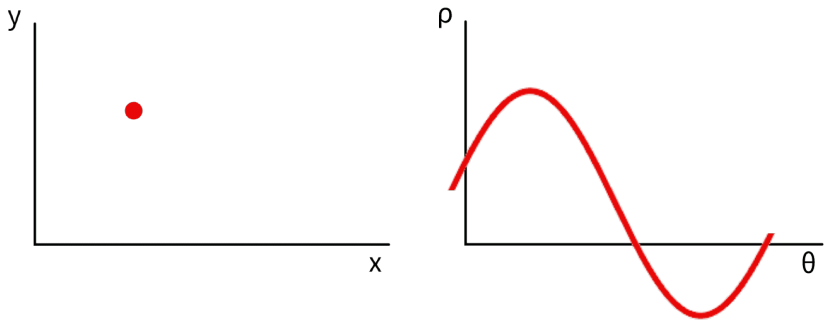
Na obrázku 3.3(a) je k vidění více přímek, které prochází jedním společným bodem a na obrázku 3.3(b) je jejich reprezentace v Houghově prostoru. Z obrázku je vidět, že body v Houghově prostoru představující přímky, tvoří sinusoidu.



(a) Přímky v obrazovém prostoru (b) Přímky v Houghově prostoru

Obrázek 3.3: Přímky procházející jedním společným bodem

Když se do rovnice $\rho = x\cos\theta + y\sin\theta$ dosadí všechny možné hodnoty parametru θ, pro konkrétní bod v obraze (v rovnici tedy budou fixní souřadnice $[x, y]$), vznikne tak v Houghově prostoru sinusoida (znázorněno na obrázku 3.4).



(a) Bod (b) Sinusoida

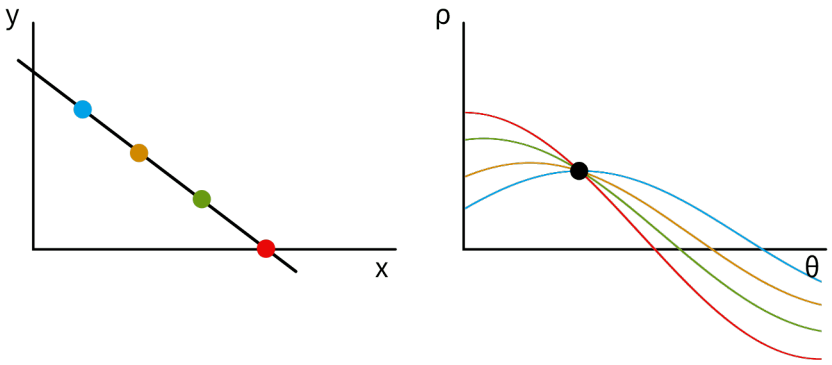
Obrázek 3.4: Bod v obrazovém a Houghově prostoru

Na obrázku 3.5 je ukázáno, jak by to vypadalo pro několik bodů ležících na přímce. Je vidět, že sinusoidy představující body se protínají právě v bodě, který představuje přímku.

3.3.3 Vyhledávání přímek

Houghova transformace pracuje s binárním obrazem, typicky se předzpracovává pomocí Cannyho hranového detektoru 3.2.

Přímky se vyhledávají tak, že se každý bod v obraze vykreslí do Houghova prostoru a v místě kde se sinusoidy nejvíce protínají vzniká kandidát na přímku [4][13].



(a) V obrazovém prostoru (b) V Houghově prostoru

Obrázek 3.5: Body ležící na přímce

3.4 Morfologické operace

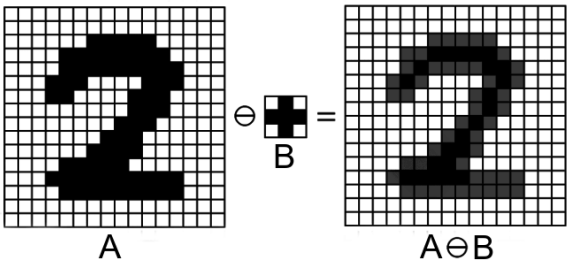
Morfologické operace jsou transformace založené na tvaru obrazu. Jsou prováděny nad binárními obrazy a potřebují dva vstupy, jedním je originální obrázek, druhý je tzv. strukturální element (také nazývaný jako jádro), který rozhoduje o způsobu operace. Základní morfologické operace jsou eroze a dilatace, další operace jsou jejich kombinace.

3.4.1 Eroze

A a B jsou množiny v prostoru Z^2 (A představuje obraz a B je strukturální element), eroze množiny A množinou B, značená $A \ominus B$, je definována jako:

$$A \ominus B = \{z|(B)_z \subseteq A\}$$

Z této rovnice vyplývá, že výsledkem eroze je množina všech bodů z, pro které platí, že posun jádra B po z je podmnožinou A. Transformace obrazu pomocí eroze je znázorněna na obrázku 3.6. Z obrázku jde vidět, že výsledkem je zmenšení černé oblasti obrazu (šedé body už nejsou součástí obrazu, jsou zde jako znázornění bodů, které byly z obrazu odebrány).



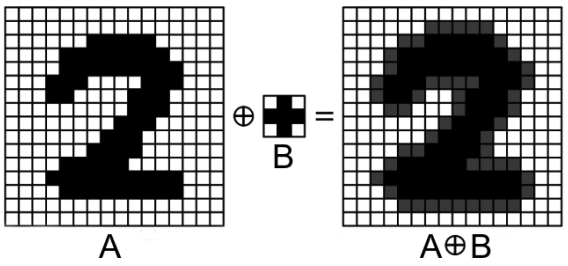
Obrázek 3.6: Binární eroze obrazu[1]

3.4.2 Dilatace

A a B jsou množiny v prostoru Z^2, dilatace množiny A množinou B, značená $A \oplus B$, je definována jako:

$$A \oplus B = \{z|(\hat{B})_z \cap A \neq \varnothing\}$$

Dochází k tomu, že pokud strukturální prvek B překrývá alespoň jeden prvek množiny A, je množina A rozšířena. Jak je vidět na obrázku 3.7, dochází k rozšiřování černých oblastí obrazu (šedé body jsou součástí výsledného obrazu).



Obrázek 3.7: Binární dilatace obrazu[2]

3.4.3 Otevření

Jedná se o proces kdy je eroze následována dilatací, tedy:

$$A \circ B = (A \ominus B) \oplus B$$

Často se používá pro odstranění šumu v obraze.

[1] Obrázek převzat z [19]
[2] Tamtéž

Obrázek 3.8: Otevření[3]

3.4.4 Uzavření

Proces opačný otevření, tzn. že se nejdříve provede dilatace a následně eroze:

$$A \bullet B = (A \oplus B) \ominus B$$

Obrázek 3.9: Uzavření[4]

Více viz [6] (odkud byly převzaty obrázky 3.9 a 3.8) a [15].

3.5 Knihovna OpenCV

OpenCV (Open Source Computer Vision Library) je knihovna nabízející funkce pro počítačové vidění a strojové učení napsaná v jazyce C a C++, díky čemuž je knihovna dobře přenositelná. Původně byla vyvíjena společností Intel. V současné době je publikována pod licencí BSD a na vývoji se podílí i uživatelé mimo Intel.

S knihovnou je možné pracovat v C/C++, Pythonu, Javě a MATLABU a podporuje Windows, Linux, Android, Mac OS a iOS [7].

OpenCV poskytuje velkou škálu funkcí pro zpracování obrazu (i videa) a má širokou komunitu. Má modulární strukturu, což znamená, že každý balíček obsahuje několik sdílených nebo statických knihoven. Jsou dostupné následující moduly:

- **Core**
  Kompaktní modul definující základní datové struktury a základní funkce používané ostatními moduly.

- **Imgproc**
  Modul pro zpracování obrazu obsahující lineární a nelineární filtry obrazu, geomet-

[3] Převzato z [6].
[4] Tamtéž.

rické obrazové transformace (změna velikosti, perspektivní a afinní transformace, ...), konverze mezi barevnými prostory, histogramy, atd.

- **Video**
  Modul pro analýzu videí nabízející predikci pohybu, subtrakci pozadí, sledování objektů, atd.
- **Calib3D**
  Kalibrace fotoaparátu/kamery a 3D rekonstrukce.
- **Features2D**
  Detektory, deskriptory.
- **Objdetect**
  Detekce objektů a instancí předdefinovaných tříd (např. obličeje, oči, lidé, auta, atd.).
- **HighGui**
  Interface pro jednoduché možnosti UI.
- **VideoIO**
  Interface pro zachytávání videa a video kodeky.
- a další pomocné moduly [8].

3.6 Tesseract

Tesseract je jeden z nejznámějších enginů pro optické rozpoznávání znaků. Byl původně vyvinut ve společnosti HP mezi lety 1985 a 1995. V roce 2006 byl vypuštěn do světa jako open source a v dnešní době je vyvíjen pod záštitou Google [23].

Tesseract podporuje kódování UTF-8 a dokáže rozpoznávat více než 100 jazyků, včetně češtiny, to ale z hlediska této práce není důležité, protože si vystačí s rozpoznáním číslic. Tesseract poskytuje vytrénovaná data a také umožňuje si vytrénovat data vlastní. Pro práci s obrazem využívá knihovnu Leptonica.

Zdrojový kód je dostupný na GitHubu[5] a je napsán v jazyce C a C++. Tesseract neposkytuje grafické uživatelské rozhraní (existuje však několik grafických uživatelkých rozhraní, která nejsou oficiální), lze s ním pracovat přes příkazový řádek nebo pomocí API, např. pro práci s Javou existuje Java JNA wrapper pojmenovaný *Tess4J*. Podporuje formáty: TIFF, JPEG, GIF, BMP a PDF.

Pro podporu Android existuje verze této knihovny nazvaná *tess-two*[6]. Pro práci s Android musí být soubor s vytrénovanými daty zkopírován do úložiště zařízení do podadresáře pojmenovaného „`tessdata`“.

3.6.1 Rozpoznávání znaků

Rozpoznávání znaků je možné provádět i na předem neupraveném obrázku, takové rozpoznávání ale není příliš přesné. Vhodnějším přístupem je znaky rozpoznávat z binárního obrazu (doporučuje se černé popředí a bílé pozadí).

Rozpoznané znaky jsou vráceny jako řetězec spolu s pravděpodobností správnosti rozpoznání (číslo 0-100).

[5] https://github.com/tesseract-ocr/tesseract
[6] https://github.com/rmtheis/tess-two

Kapitola 4

Umělá inteligence

Pojem umělá inteligence (artificial intelligence) byl ustálen, když v roce 1956 John McCarthy uspořádal konferenci na Darthmouth College, která měla toto sousloví ve svém názvu - „The Darthmouth Summer Research Project on Artifical Intelligence“. Zmínky o umělé inteligenci jsou však starší než její pojmenování, příkladem je třeba pověst o pražském golemovi nebo příběhy z řecké mytologie [21].

Přesto, že se jedná o poměrně starý pojem, jeho definice je stále problematická a doteď nebyla stanovena definice, jež by byla přesná. Je to hlavně proto, že neexistuje přesná definice inteligence. Inteligence bývá přisuzována lidem, ale také zvířatům. Z pohledu této práce bude umělá inteligence chápána jako vědní disciplína.

Marvin Minsky definoval umělou inteligenci v roce 1967: „Umělá inteligence je věda o vytváření strojů nebo systémů, které budou při řešení určitého úkolu užívat takového postupu, který, kdyby ho dělal člověk, bychom považovali za projev inteligence.“ [17]

4.1 Základní pojmy

V této části jsou vysvětleny základní pojmy související s řešením úloh pomocí umělé inteligence, které jsou potřebné pro popsání metod použitých v této práci.

Stavový prostor

Stavový prostor je orientovaný graf sestávající z uzlů, které představují jednotlivé stavy řešené úlohy. Hrany mezi uzly představují přechody mezi těmito stavy, ke kterým dochází operacemi předem definovaných operátorů. Stavový prostor je tedy definován dvojicí (S, O), kde S značí množinu všech možných stavů a symbol O množinu operátorů, kterými lze měnit stavy.

Úloha

Úloha je definována jako dvojice (s_0, G), kde s_0 označuje počáteční stav úlohy a $G \subset S$ množinu cílových stavů dané úlohy, tedy řešení. Množina G nemusí být nutně jednoprvková (úloha může mít více správných řešení), ale často jednoprvková je.

Řešení úlohy

Řešení úlohy je posloupnost operátorů $s_1 = o_1(s_0), s_2 = o_2(s_1), \ldots, s_n(s_(n-1)), s_n \in G$, tedy cesta od počátečního stavu k řešení.

4.2 Řešení úloh prohledáváním stavového prostoru

Aby bylo možné najít řešení úlohy, je třeba prohledávat stavový prostor. Existuje řada metod, které toto umožňují (např. prohledávání do hloubky, prohledávání do šířky, obousměrné prohledávání, A* a další). Metody řešení úloh prohledáváním stavového prostoru jsou hodnoceny podle následujících vlastností:

- **Úplnost**
  Metoda je úplná, pokud je pomocí této metody možné úlohu vyřešit (v případě, že řešení existuje).
- **Optimálnost**
  Metoda je optimální, když nalezne nejlepší možné řešení.
- **Časová složitost**
  Čas potřebný pro vyřešení úlohy danou metodou. Čas je závislý na výpočetním výkonu počítače, který výpočet provádí, proto se používá počet prozkoumaných stavů.
- **Paměťová složitost**
  Množství operační paměti potřebné pro vyřešení úlohy. Používá se počet stavů současně uchovaných v paměti.

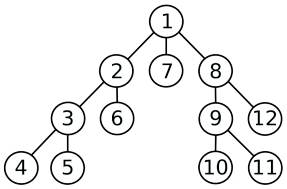
V následující části textu je blíže popsána metoda prohledávání do hloubky, jejíž pochopení je třeba pro porozumění backtrackingu, který výsledná aplikace používá.

4.2.1 Prohledávání do hloubky

Prohledávání do hloubky, anglicky Depth-first search (DFS), je neinformovaná metoda, jež rozšiřuje uzel, který je přímým potomkem zkoumaného uzlu. Tímto způsobem rozšiřuje stavový prostor, dokud nenarazí na uzel, který je řešením, nebo na uzel, který nemá žádné potomky. V případě, že narazí na uzel bez potomků, dojde k navrácení do uzlu, který ještě nebyl rozšířen o své potomky.

Na obrázku 4.1 je ukázka, jak by mohlo vypadat prohledávání stavového prostoru touto metodou. Metoda má následující vlastnosti (b je faktor větvení, m je maximální délka cesty):

- není úplná,
- není optimální,
- časová složitost: b^m,
- paměťová složitost: m.



Obrázek 4.1: Postup prohledávání uzlů metodou DFS

4.3 Úlohy s omezujícími podmínkami

Jedná se o úlohy, kdy není důležitá posloupnost operátorů vedoucí k řešení, ale pouze cílový stav (řešení), který musí splňovat předem dané podmínky. Anglicky se těmto úlohám říká Constraint Satisfaction Problems (CSP). Do této skupiny úloh patří i Hitori, kde jsou omezujícími podmínkami tři pravidla zmíněná v kapitole 2.

Úlohy s omezujícími podmínkami lze definovat následovně. Je dána množina proměnných, konečná a diskrétní doména pro každou proměnnou a množina podmínek. Každá podmínka je definována nad podmnožinou množiny proměnných a limity kombinací hodnot, kterých dané proměnné nabývají. V počátečním stavu nemá žádná proměnná přiřazenou hodnotu. Cílem je najít takové přiřazení hodnot proměnným, které splňuje všechny podmínky. [18]

Úlohy s omezujícími podmínkami je možné řešit pomocí metod prohledávání stavového prostoru. Existují ale i speciální metody, které jsou výrazně efektivnější. Jednou z nich je například backtracking.

4.3.1 Backtracking

Backtracking (metoda zpětného navrácení) je algoritmus vycházející z prohledávání do hloubky, nedochází ale ke generování všech uzlů. V případě že přiřazením do proměnné dojde k porušení některé z podmínek, dojde k navrácení do předchozího uzlu stavového prostoru. Díky tomu má algoritmus nízkou paměťovou náročnost. Časová náročnost je exponenciální. Shrnutí vlastností metody (n je počet proměnných, m je maximální počet přiřaditelných hodnot):

- je úplná,
- optimální (každé řešení CSP je optimální),
- časová náročnost: n,
- paměťová náročnost: m^n

Více o umělé inteligenci lze najít v [20].

Kapitola 5

Android

Android je mobilní operační systém (OS) založený na linuxovém jádře, který je dostupný jako otevřený systém (open source). Vývoj je veden společností *Google* pod hlavičkou organizace *Open Handset Alliance*. Systém je navržen primárně pro dotykové obrazovky, ale používá se např. i u chytrých televizí.

První Android vznikl v roce 2008 a od té doby bylo vydáno celkem 30 verzí. V současné době je poslední verzí Android 11. Tento OS využívá 72.48% mobilních zařízení (k prosinci 2020), to jej řadí na první místo. Druhým v pořadí je operační systém *iOS* od společnosti *Apple* s podílem 26.91% [5].

Oficiálním vývojovým prostředím pro aplikace běžící na tomto OS je Android Studio, které poskytuje emulátory mobilních telefonů, tabletů, chytrých hodinek a dalších.

5.1 Architektura operačního systému

OS Android je založen na linuxovém jádře (jádro je modifikované). Architektura OS Android sestává z 5 vrstev:

- **Linuxové jádro**
  Je nejnižší vrstvou architektury. Závisí na něm např. Android Runtime pro základní funkce jako správa vláken nebo nízkoúrovňová správa paměti.
- **Hardware Abstraction Layer (HAL)**
  Poskytuje abstraktní vrstvu mezi zařízením a hardwarem. Skládá se z několika knihoven modulů, které implementují rozhraní pro specifický typ hardwaru (např. modul fotoaparátu nebo bluetooth).
- **Knihovny a Android Runtime**
  Nativní knihovny Androidu jsou napsány v C/C++. Android Runtime slouží pro běh aplikací (aplikace jsou psány v Javě a je potřeba je přeložit do nativního kódu).
- **Java API Framework**
  Nad touto vrstvou běží samotné aplikace. Poskytuje služby zpřístupňující data v jiných aplikacích, služby umožňující běh aplikací na pozadí a další.
- **Systémové aplikace**
  Mezi systémové aplikace patří např. SMS zprávy, kalendář, internetový prohlížeč, ...

Jednotlivé vrstvy mezi sebou spolupracují. Více viz [9].

5.2 Verze Androidu

Starší verze systému mají názvy podle sladkostí seřazeny podle abecedy. U verze 10 se Google rozhodl zjednodušit způsob značení verzí, systém proto nenese název začínající písmenem Q, ale pouze číselné označení. Přehled verzí Androidu v závislosti na API (Application Software Interface) je v tabulce 5.1.

Verze	Datum uvedení	API úroveň
Android 1.0 Apple Pie	23. září 2008	API 1
Android 1.1 Banana Bread	9. února 2009	API 2
Android 1.5 Cupcake	27. dubna 2009	API 3
Android 1.6 Donut	15. září 2009	API 4
Android 2.0 Eclair	26. října 2009	API 5
Android 2.0.1 Eclair	3. prosince 2009	API 6
Android 2.1 Eclair	12. ledna 2010	API 7
Andoroid 2.2.x Froyo	20. května 2010	API 8
Android 2.3 - 2.3.2 Gingerbread	6. prosince 2010	API 9
Android 2.3.3 - 2.3.7 Gingerbread	9. února 2011	API 10
Android 3.0 Honeycomb	22. února 2011	API 11
Android 3.1 Honeycomb	10. května 2011	API 12
Android 3.2.x Honeycomb	15. července 2011	API 13
Android 4.0. - 4.0.2 Ice Cream Sandwich	18. října 2011	API 14
Android 4.0.3 - 4.0.4 Ice Cream Sandwich	16. prosince 2011	API 15
Android 4.1.x Jelly Bean	9. července 2012	API 16
Android 4.2.x Jelly Bean	13. listopadu 2012	API 17
Android 4.3.x Jelly Bean	24. července 2013	API 18
Android 4.4 - 4.4.4 KitKat	31. října 2013	API 19
Android 5.0 Lollipop	12. listopadu 2014	API 21
Android 5.1 Lollipop	9. března 2015	API 22
Android 6.0 Marshmallow	5. října 2015	API 23
Android 7.0 Nougat	22. srpna 2016	API 24
Android 7.1 Nougat	4. října 2016	API 25
Android 8.0.0 Oreo	21. srpna 2017	API 26
Android 8.1.0 Oreo	5. prosince 2017	API 27
Android 9 Pie	6. srpna 2018	API 28
Android 10	3. září 2019	API 29
Androd 11	8. září 2020	API 30

Tabulka 5.1: Verze Androidu [1]

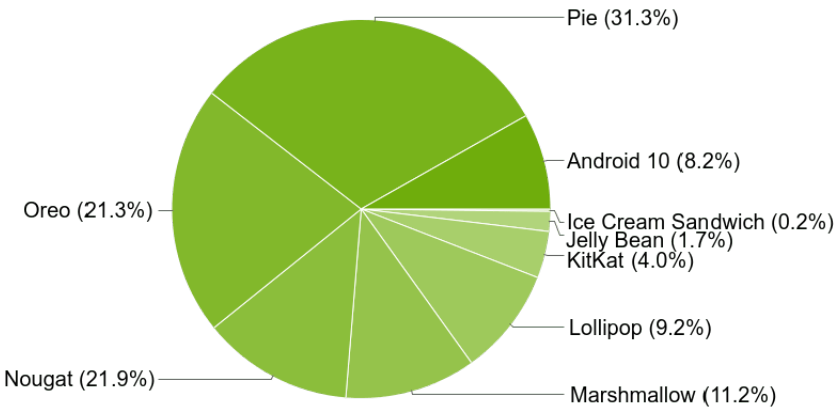
5.2.1 Zastoupení verzí Android

Pro výběr API, pod kterou se vývojář rozhodne svou aplikaci vyvíjet, je důležité znát, které verze se aktuálně využívají. Tuto informaci Google dříve poskytoval na webu. V současné době zde tato informace k dispozici není. Místo toho je v Android Studiu při vytváření nového projektu možné zjistit kumulativní distribuci verzí (náhled toho jak zobrazení v An-

droid Studiu vypadá je na obrázku 5.1). Z toho lze spočítat zastoupení jednotlivých verzí Android (znázorněno na obrázku 5.2) [2].

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99.8%
4.2 Jelly Bean	17	99.2%
4.3 Jelly Bean	18	98.4%
4.4 KitKat	19	98.1%
5.0 Lollipop	21	94.1%
5.1 Lollipop	22	92.3%
6.0 Marshmallow	23	84.9%
7.0 Nougat	24	73.7%
7.1 Nougat	25	66.2%
8.0 Oreo	26	60.8%
8.1 Oreo	27	53.5%
9.0 Pie	28	39.5%
10. Android 10	29	8.2%

Obrázek 5.1: Kumulativní distribuce verzí



Obrázek 5.2: Koláčový graf reprezentující podíl využití verzí androidu[1]

5.3 Aktivita

Třída `Activity` je hlavní komponentou aplikace. Aktivitu je možné implementovat jako podtřídu třídy `Activity`. Aktivita poskytuje okno, do kterého aplikace vykreslí uživatelské

[1]Obrázek převzat z [2].

rozhraní. Většina aplikací se skládá z několika aktivit, z nichž jedna je hlavní (ta která se načte jako první při otevření aplikace).

5.3.1 Životní cyklus aktivity

V průběhu svého životního cyklu prochází aktivita několika stavy. Ke zpracování přechodů mezi stavy se používají callback metody. Callback metody jsou následující:

- **onCreate()**
  Tato callback metoda musí být implementována. Je volána, když systém vytvoří aktivitu (začátek životního cyklu). Když je `onCreate()` dokončena, následuje `onStart()`.
- **onStart()**
  Aktivita je viditelná pro uživatele a stává se interaktivní.
- **onResume()**
  Tato metoda je volána na začátku životního cyklu aktivity a pokud byla aktivity na pozadí a právě přechází do popředí – tj. po `onPause()`.
- **onPause()**
  Volána, pokud je aktivita dána do pozadí.
- **onStop()**
  Aktivita je zastavena např. když začíná nová aktivita nebo daná aktivita přechází do stavu `Resumed`. Následuje `onRestart()` nebo `onDestroy()`.
- **onRestart()**
  Metoda je volána, pokud je aktivita ve stavu `Stopped` a chystá se restartovat. Obnoví se stav aktivity, ve kterém byla předtím než byla zastavena (ve stavu Stopped).
- **onDestroy()**
  Volána při ukončení aktivity. Obvykle je implementována proto, aby bylo zajištěno uvolnění zdrojů aktivity.

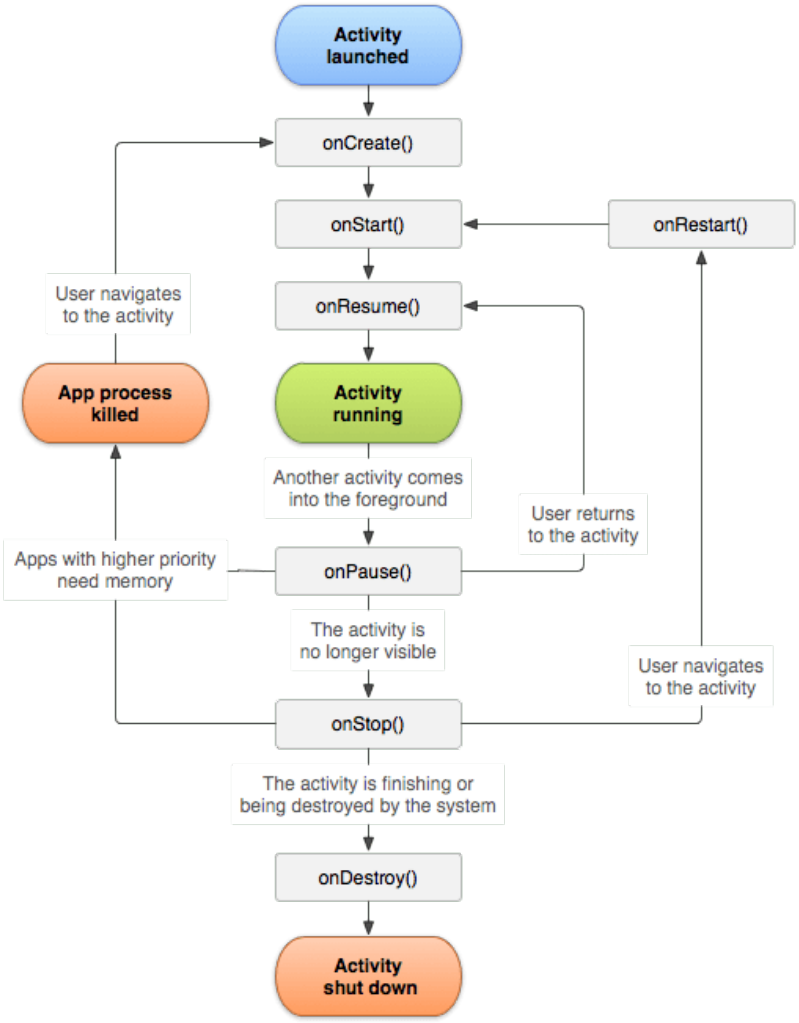
Správná implementace callback metod zajišťuje, že aplikace nespadne třeba po přepnutí do jiné aplikace, nebo že uživatel nepřijde o svůj postup ve hře. Na obrázku 5.3 je k vidění ilustrace životního cykly aktivity. Více viz [11] odkud byl převzat i obrázek.

Fragment

Frament je součástí aktivity, dalo by se říct, že je to podaktivita. Jedna aktivita může obsahovat více fragmentů, fragment ale nemůže existovat bez aktivity. Životní cyklus fragmentu je závislý na životním cyklu aktivity, do které fragment patří. Fragment má oproti aktivitě navíc callback metodu **onAttach()**, která je volána, když je fragment připojen k aktivitě.

5.4 Databáze SQLite

SQLite je relační databázový systém obsažený v knihovně napsané v jazyce C. SQLite slouží k uložení perzistentních dat aplikace do databáze a pro přístup k těmto datům. Perzistentní data jsou data, která jsou trvale uložena a přežívají i ukončení nebo pád aplikace.



Obrázek 5.3: Ilustrace životního cyklu aktivity[2]

Podporuje primitivní datové typy (Integer, String, Double a Byte), pro uložení jiných datových typů slouží konvertory, které poskytuje knihovna Room 5.4.1. Knihovna Room je v dokumentaci Android doporučena pro práci s SQLite databází.

5.4.1 Room

Knihova Room je součástí Android Jetpack a umožňuje vytvoření a manipulaci s databází SQLite v jazyce Java. Poskytuje abstraktní vrstvu mapující entity databáze do objektů POJO (Plain Old Java Objects). Velkou výhodou je validace SQL dotazů při překladu. Architektura této knihovny obsahuje tři hlavní komponenty popsány níže a znázorněny na obrázku 5.4. Jsou to komponenty:

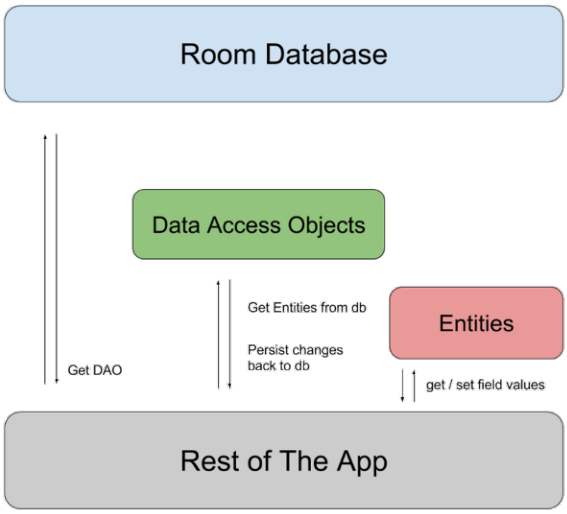
- **Entity**
  Pro každou třídu s anotací `@Entity` Room vytvoří tabulku v databázi. Každá členská proměnná v takové třídě představuje sloupec dané tabulky v databázi. Pro de-

[2]Obrázek převzat z [11].

klaraci unikátního primárního klíče slouží anotace `@PrimaryKey`. Pomocí anotace `@ColumnInfo` je možné přejmenovat sloupec a toto jméno používat v SQL dotazech. Jinak se používá jméno proměnné, tak jak je uvedeno ve třídě.

- **Data Access Object (DAO)**
  Tato komponenta poskytuje abstraktní rozhraní definující metody pro komunikaci s databází. Dotazy mohou být vytvořené explicitně pomocí anotací `@Insert`, `@Delete` a `@Update`, nebo lze vytvořit vlastní dotazy pomocí anotace `@Query` s argumentem obsahujícím dotaz v syntaxi jazyka SQL.

- **Database**
  Tato třída slouží pro vytvoření samotné databáze, musí dědit od třídy `RoomDatabase`. Značí se anotací `@Database` a pomocí argumentů jsou předány třídy `Entity`, tedy tabulky, které databáze obsahuje. Třída `Database` poskytuje aplikaci instance objektů `DAO` přidružené této databázi.



Obrázek 5.4: Diagram architektury knihovny Room[3]

Vice viz [10] odkud je převzat i obrázek 5.4.

5.5 Google Play

Google Play je online distribuční služba poskytující kromě aplikací pro zařízení Android také e-knihy, filmy a hudbu. Funguje od roku 2008. Jeho předchůdcem je Android Market.

Pro publikování vlastní aplikace na Google Play je potřeba vytvořit účet pro vývojáře a uhradit jednorázový registrační poplatek 25$. Na Google Play je možné vložit dva formáty vydání: bundle (.aab) a APK (.apk), je vyžadován unikátní název balíčku. Maximální velikost aplikace je v současné době 150 MB.

Vydání aplikace je možné vytvořit ve třech druzích testovacích kanálů nebo v produkčním kanálu. Druhy testovacích kanálů jsou:

[3]Obrázek převzat z [10].

- **Otevřené testování**
  Otevřené testovací vydání je dostupné všem registrovaným testerům na Google Play.
- **Uzavřené testování**
  Uzavřené testování umožňuje oproti internímu testování přidat kanál pro zaslání zpětné vazby.
- **Interní testování**
  Interní testovací vydání jsou dostupná maximálně 100 testerům vlastního výběru. Je potřeba uvést seznam testerů (jejich gmailové adresy).

Produkční vydání jsou dostupná všem uživatelům (nejen testerům) na Google Play.

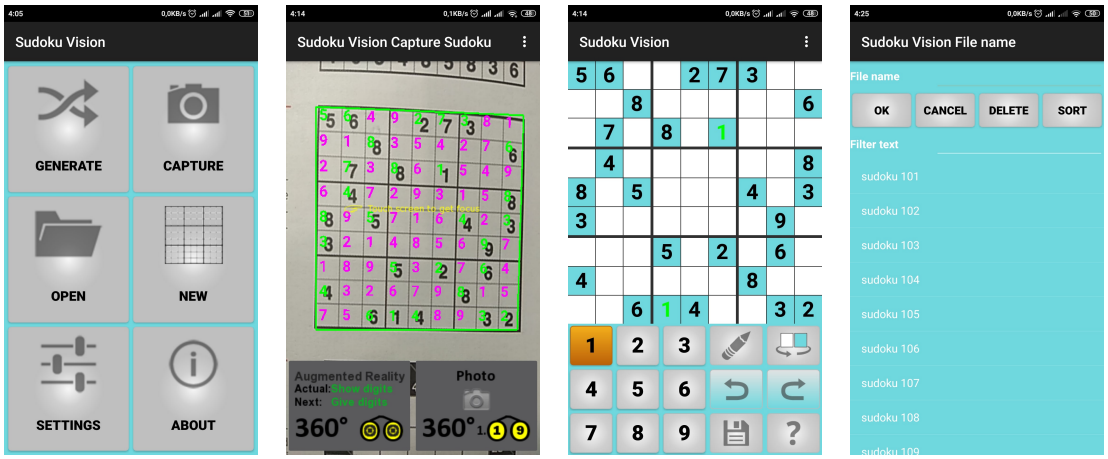
Kapitola 6

Zhodnocení podobných existujících řešení

Aplikace, která by nabízela možnost naskenování hry Hitori a její dohrání zatím žádná neexistuje. Na Google Play jsou dostupné aplikace, které nabízí možnost řešení Hitori s databází her. Jednou z takových aplikací je Hitori. Dostupné jsou také aplikace pro naskenování podobných logických her s čísly (jako je Sudoku). Zástupcem těchto aplikací je Sudoku Vision.

6.1 Aplikace Sudoku Vision

Sudoku Vision od vývojáře Rogerlebo má na Google Play[1] hodnocení 4.5/5. Kromě zachycení Sudoku z fotografie nabízí generování nových her a manuální vytvoření vlastního zadání. Je k dispozici nápověda, kde si uživatel může zobrazit, která čísla je možné do nějaké buňky doplnit. A nebo uživatel může aplikaci nechat, aby číslo doplnila za něj.



(a) Hlavní menu (b) Skenování Sudoku (c) Řešení Sudoku (d) Nabídka uložených her

Obrázek 6.1: Snímky obrazovky z aplikace Sudoku Vision

[1] https://play.google.com/store/apps/details?id=com.rogerlebo.sudokuvision

Zhodnocení

Uživatelské rozhraní není moc přehledné, je až matoucí. Design nevypadá moc dobře ani vzhledem k tomu, že poslední aktualizace je z roku 2017. Sudoku je možné naskenovat pouze živě, chybí možnost skenování již pořízené fotografie. Rozpoznání čísel funguje spolehlivě.

6.2 Aplikace Hitori

Hitori je aplikace, kterou vyvíjí Apo-Games a na Google Play[2] má hodnocení 4.6/5. Snímek obrazovky ze hry je na obrázku 6.2. Aplikace rozlišuje smazané a ponechané buňky barevně. Když uživatel udělá chybu, která porušuje pravidla hry, je na to upozorněn červeným zbarvením číslic v buňkách, jež pravidla porušují. Na obrázku 6.2 je znázorněno porušení všech tří pravidel Hitori (viz kapitola 2).

Aplikace zobrazuje časomíru, nabízí tlačítka pro restart hry, nápovědu, navrácení o krok zpět a tlačítko pro zavření hry (návrat do menu). Navíc aplikace v hlavním menu obsahuje položku pro zobrazení návodu, jak hru hrát. Označování polí je prováděno jedním kliknutím pro smazání (hnědé zbarvení), dalším kliknutím označení pro ponechání pole (žlutá barva), dalším kliknutím je pole uvedeno do původního stavu.

Obrázek 6.2: Snímek obrazovky z aplikace Hitori

Zhodnocení

Rozmístění tlačítek není úplně šťastné. Tlačítko pro zavření hry je zbytečné, protože stejnou funkci poskytuje systémové tlačítko zpět. Navíc zabírá pozici, která by byla vhodná spíš pro tlačítka, která uživatel využívá nejčastěji, jelikož jsou blízko palci pravé ruky.

[2] https://play.google.com/store/apps/details?id=org.apogames.hitori

6.3 Shrnutí

V režimu hry by měla výsledná aplikace určitě zobrazovat porušení pravidel hry. Zvýraznění číslic červenou barvou se zdá být dobrým řešením. Změnu stavů polí opakovaným klikáním bude aplikace také používat. Je tak potřeba méně tlačítek a je to intuitivní. Také by měl být měřen čas a měla by být k dispozici tlačítka pro restart hry, nápovědu a navrácení o krok zpět. Jen by tlačítka měla být lépe uspořádána, aby byla lépe dostupná.

Aplikace by měla podporovat naskenování hry i ze staré fotografie. Uživatelské rozhraní musí být o dost přehlednější než je tomu u aplikace Sudoku Vision. Design by měl být výrazně čistší než u obou výše popsaných aplikací.

Kapitola 7

Návrh aplikace

Po prozkoumání podobných existujících aplikací byla aplikace navržena tak, aby v ní uživatel našel vše co potřebuje a nic zbytečného navíc. V této části je popsán základní návrh podoby grafického uživatelského rozhraní. Také je zde popsáno, jaké informace se budou ukládat v databázi a jakým způsobem bude hledáno celkové řešení naskenovaného zadání Hitori, ale i jakým způsobem jsou uživateli nabízeny jednotlivé kroky, které mu bude poskytovat nápověda.

7.1 Databáze her

Aby bylo možné se vracet k rozehraným hrám, je k dispozici databáze her. Do databáze se ukládají všechny hry, které uživatel naskenuje. V databázi se o hře ukládají následující informace:

- identifikační číslo hry,
- čas strávený ve hře,
- datum vytvoření,
- hrací pole,
- fotografie hracího pole s transformovanou perspektivou pro náhled,
- a správné řešení.

7.2 Návrh grafického rozhraní

Výsledná aplikace by měla mít přehledné a intuitivní uživatelské rozhraní. Grafický vzhled je první, s čím uživatel přijde do styku. Proto je žádoucí, aby uživatele neodradil hned po prvním použití aplikace.

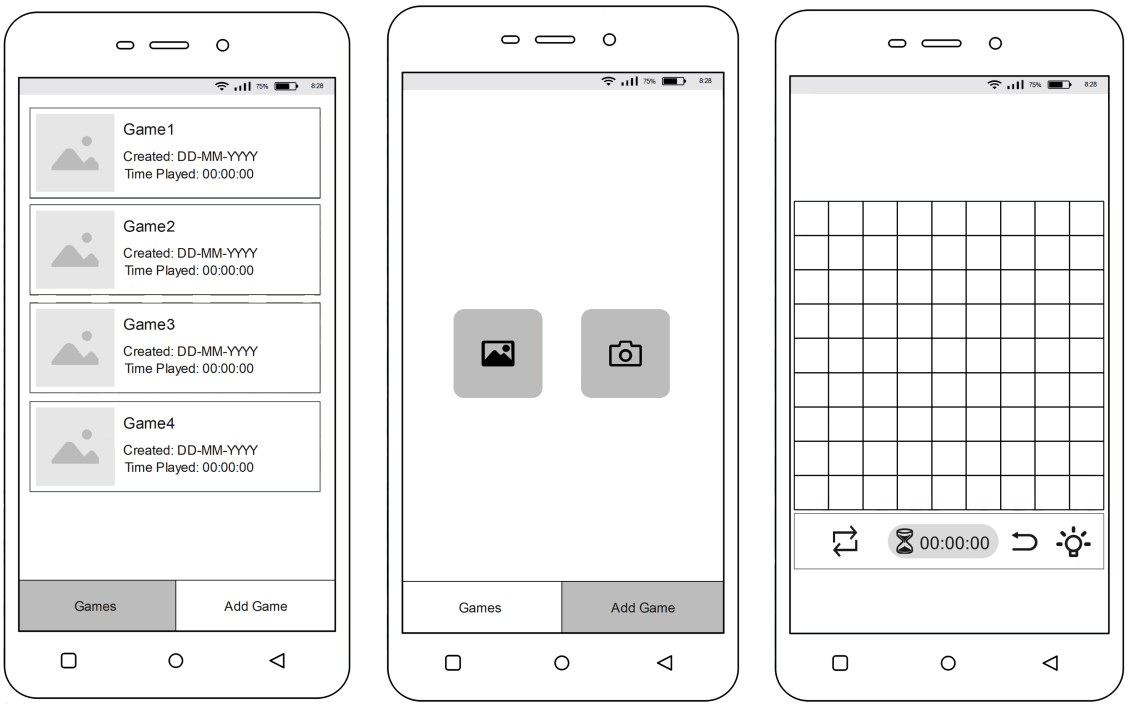
Hlavní stránka, do které se uživatel dostane hned po spuštění aplikace, je rozdělena na dvě části. První část zobrazuje seznam uložených her, druhá část umožňuje nahrání fotografie se zadáním hry následované skenováním. Mezi těmito částmi je možné přepínat pomocí spodní navigační lišty.

Jednotlivé položky v seznamu her tvoří: název hry, datum vytvoření, čas hry a náhled fotografie, ze které byla hra naskenována, viz obrázek 7.1(a). Při větším množství položek bude možné scrollovat.

Pro přidání nové hry je třeba nahrát fotografii s hracím polem. To je provedeno vybráním fotografie z galerie nebo vyfocením, návrh obrazovky nabízející tyto možnosti je na obrázku 7.1(b).

Po naskenování hry je okamžitě otevřeno prostředí pro její vyřešení. Při řešení Hitori je k dispozici náhled s časomírou a tlačítka usnadňující úspěšné vyřešení zadání. Podle pořadí v obrázku 7.1(c) jsou to tlačítka s funkcemi: restart hry, návrat o krok zpět, nápověda. Tlačítka jsou umístěna dole pod hracím polem, aby na ně uživatel pohodlně dosáhl palcem pravé ruky. Nejblíže má tlačítko pro nápovědu a návrat o krok zpět, u těchto tlačítek se předpokládá, že budou využívána častěji než restart hry.

Hrací pole je ovládáno stejně jako v aplikaci Hitori od Apo-Games 6.2, tedy opakovaným klikáním na jedno pole se toto pole postupně smaže, ponechá a následně uvede do původního stavu. Pokud uživatel udělá ve hře chybu, je na to upozorněn červeným zbarvením. Když je hra dokončena, obarví se všechna neoznačená pole, jako ponechaná. Při dokončení hry by se měla taky zastavit časomíra.



(a) Seznam uložených her (b) Přidání nové hry (c) Řešení Hitori

Obrázek 7.1: Návrh grafického rozhraní

7.3 Naskenování Hitori

Konceptuální návrh skenování Hitori sestává z několika kroků, které je nutné provést:

1. Nalezení hracího pole ve fotografii.
2. Určení velikosti hracího pole.
3. Extrakce číslic z hracího pole.

4. Rozpoznání číslic.

5. Vytvoření objektu reprezentujícího hrací pole.

7.4 Hledání správného řešení

Před spuštěním hry je potřeba zjistit, jestli má dané zadání vůbec řešení. Správné řešení je hledáno kombinací backtrackingu a technik řešení uvedených v části 2.1. Nejdříve jsou provedeny techniky řešení v následujícím pořadí:

1. Provede se operace ponechání buněk kolem smazaných buněk (v počátečním stavu se neprovede nic).

2. Hledá se dvojice buněk se stejnou hodnotou, které od sebe dělí jen jedna buňka a tato oddělující buňka je ponechána.

3. Aplikuje se technika indukce párů.

Přitom kdykoliv je nějaká buňka smazána, jsou ponecháni její sousedi. A kdykoliv je nějaká buňka ponechána, jsou smazány další výskyty stejné hodnoty ve sloupci a řádku.

Když nastane situace, kdy se pomocí těchto postupů přestane měnit stav hracího pole a hra ještě není vyřešená, jsou vyhledány unikátní buňky a jsou ponechány. Potom přichází na řadu backtracking. Dojde k vyhledání všech neoznačených polí a pokusu tato pole smazat za podmínek: nesmazaná pole musí být vzájemně propojená, smazaná pole nesmí sousedit. Tyto dvě podmínky společně definují legální stav. Řešení je splnění těchto dvou podmínek a navíc podmínky neopakujících se hodnot v řádcích a sloupcích. Tento přístup je inspirován článkem [12].

Správné řešení je uloženo do databáze a je využíváno pro nápovědu jednoho kroku.

7.5 Hledání jednoho kroku řešení

Pokud uživatel neví jak ve hře pokračovat, má možnost využít nápovědu. Správné řešení sice aplikace zná, ale účelem je, aby uživatel dostal nápovědu, která mu pomůže přijít na další krok sám. Proto by nápověda měla pracovat se současným stavem hracího pole.

Na současný stav hracího pole jsou aplikovány techniky řešení podobně jako je tomu u hledání správného řešení. Rozdíl je ale v tom, že je změněn stav pouze jedné buňky. Jestliže už nejsou tyto techniky postačující, nehledá se další krok pomocí backtrackingu, ale využije se správné řešení uložené v databázi. V databázi se vyhledá buňka, která je v současném stavu hracího pole neoznačená, ale ve správném řešení je smazaná.

Kapitola 8

Implementace

Implementace je rozdělena do několika logických celků. Prvním z nich je naskenování Hitori z fotografie, týká se tedy zpracování obrazu. Druhým logickým celkem je hledání řešení pomocí metod umělé inteligence. Poslední část je celková aplikace.

8.1 Zpracování obrazu

Výsledkem této části jsou rozpoznané číslice a jejich pozice v herním poli. Celý proces je rozdělen do osmi mezikroků znázorněných na obrázku 8.1.

Obrázek 8.1: Diagram znázorňující postup zpracování fotografie

V prvních sedmi krocích hraje hlavní roli knihovna OpenCV (viz 3.5) a jsou implementovány ve třídě `ImageScanner`, poslední krok je proveden pomocí knihovny Tesseract (viz 3.6) ve třídě `TesseractOCR`. Jednotlivé kroky jsou podrobněji popsány v následujících podsekcích.

8.1.1 Předzpracování obrázku

Předtím, než je aplikován Cannyho hranový detektor, je potřeba obrázek připravit tak, aby bylo dosaženo co nejlepších výsledků:

1. **Převedení obrázku do stupňů šedé**
   Princip je popsán v sekci 3.1. Provedeno pomocí funkce `cvtColor()` s parametrem `COLOR_RGB2GRAY`.

2. **Provedení Gaussova rozostření**
   Princip popsán v sekci 3.2.1. OpenCV nabízí funkci `GaussianBlur()`, vstupní obrázek

je ve stupních šedé. Použitá velikost konvolučního jádra je 5×5, směrodatná odchylka je 1.

3. **Adaptivní prahování**
   Podrobněji popsáno v sekci 3.1. Provedeno pomocí funkce `adaptiveThreshold()` s hodnotou maximálního prahu 255, velikostí okolí zpracovávaného pixelu 25, střední hodnotou 1 a adaptivní metodou `ADAPTIVE_THRESH_GAUSSIAN_C`, která určuje hodnotu prahu podle váženého součtu hodnot okolí. Použitý parametr `THRESH_BINARY_INV` určuje požadovanou podobu výstupu (černé pozadí, bílé popředí). Ukázka výstupu je na obrázku 8.2(b).

8.1.2 Cannyho hranový detektor

Princip tohoto detektoru je popsán v sekci 3.2. OpenCV poskytuje funkci `Canny()`, použitý minimální práh je 50 a maximální práh je 150 (podle doporučení v dokumentaci OpenCV je třikrát větší než minimální práh). Velikost Sobelova operátoru je 5×5. Výstupem je obrázek s vyznačenými hranami, ukázka na obrázku 8.2(c).

8.1.3 Nalezení hracího pole

Pro nalezení hracího pole se využívá výstup Cannyho detektoru, kterým je obrázek s černým pozadím obsahující bíle vyznačené hrany. Tento výstup se použije jako vstup funkce vyhledávající kontury `findContours()`. Je využit mód pro vyhledávání kontur `RETR_EXTERNAL`, který zajišťuje vyhledávání pouze vnějších kontur. Dále používám metodu aproximace kontur `CHAIN_APPROX_SIMPLE`, díky které jsou na výstupu pouze přibližné kontury. Ukázka všech nalezených kontur je na obrázku 8.2(d). V dalším kroku je tedy potřeba hledanou konturu aproximovat lépe. Výstupem je seznam nalezených kontur a matice nesoucí informace o topologii obrázku (obsahuje počet položek stejný jako počet nalezených kontur).

Po nalezení kontur se v seznamu s těmito konturami najde kontura s největším obsahem, o té se předpokládá, že je hracím polem. Jak je zmíněno výše, je potřeba tuto konturu lépe aproximovat. Proto je využita funkce `approxPolyDP()`, která vrací přesnější souřadnice řídících bodů. Nalezená kontura s největším obsahem je ukázána na obrázku 8.2(e).

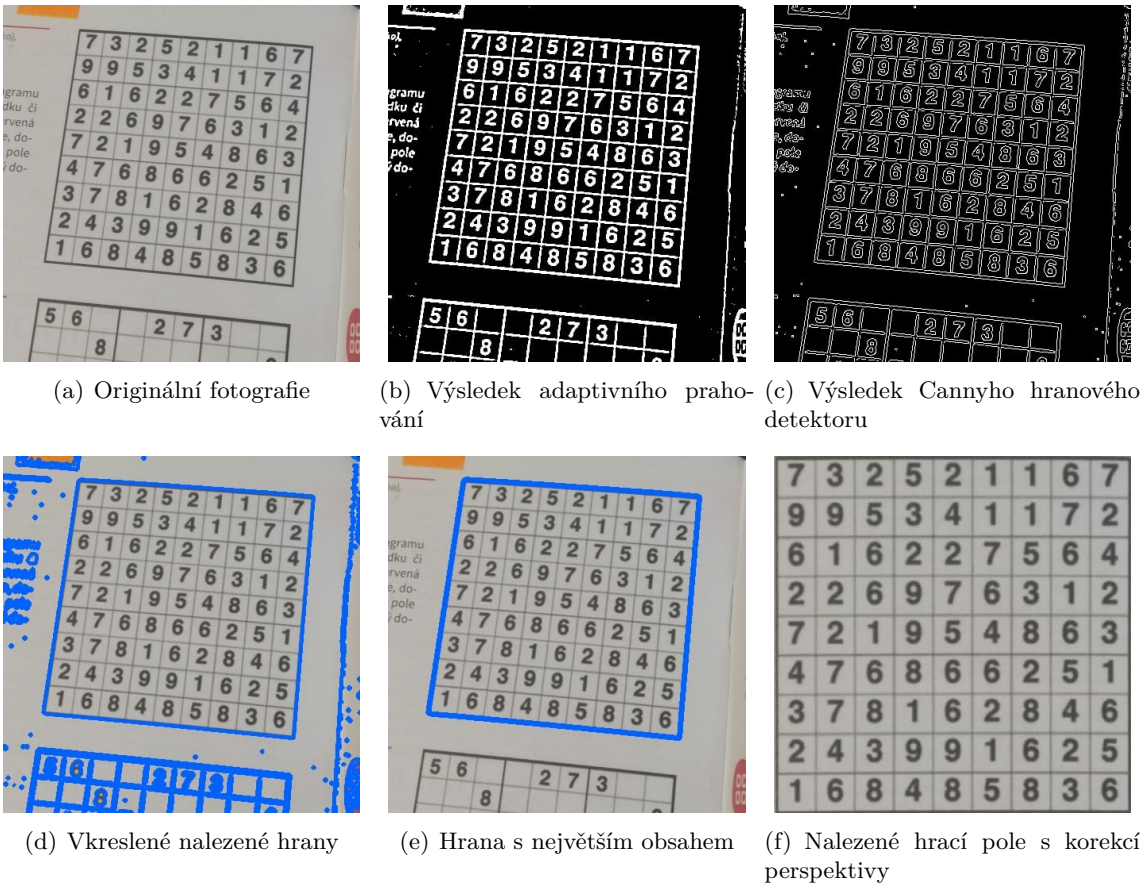
8.1.4 Transformace perspektivy

Po nalezení hracího pole je potřeba transformovat perspektivu, aby byl obrázek zbaven nepodstatných oblastí a aby byl obrázek srovnán, což je potřebné pro další zpracovávání. K tomu jsou užitečné funkce `getPerspectiveTransform()` a `warpPerspective()`.

Pro použití funkce `getPerspectiveTransform()` je potřeba seřadit body nalezené největší kontury do pořadí, se kterým tato funkce pracuje, tj. pořadí: horní levý roh, pravý horní roh, levý dolní roh, pravý dolní roh. Horní levý roh lze určit jako nejmenší součet souřadnic, pravý horní roh jako minimální rozdíl souřadnic, levý dolní roh pak jako maximální rozdíl a nakonec pravý dolní roh jako maximální součet. Výsledek transformace je k vidění na obrázku 8.2(f).

8.1.5 Zjištění velikosti hracího pole

Pro zjištění velikosti hracího pole je potřeba v obrázku nalézt přímky, které tvoří mřížku hracího pole. To je provedeno pomocí Houghovy transformace, jejíž princip je popsán v před-



(a) Originální fotografie

(b) Výsledek adaptivního prahování

(c) Výsledek Cannyho hranového detektoru

(d) Vkreslené nalezené hrany

(e) Hrana s největším obsahem

(f) Nalezené hrací pole s korekcí perspektivy

Obrázek 8.2: Mezikroky potřebné pro detekci hracího pole z fotografie

chozí kapitole 3.3. Před provedením Houghovy transformace je aplikován Cannyho hranový detektor.

V OpenCV tuto transformaci poskytuje funkce `HoughLines()`, která je volána s argumenty $\rho = 1$, $\theta = \pi/180$ a `threshold` $= 150$. Následně jsou rozlišeny vertikální a horizontální přímky. Za vertikální přímky jsou považovány takové přímky, kde $\theta = 0$, a horizontální přímky jsou takové přímky, kde body `p1` a `p2` mají shoudnou souřadnici y. Souřadnice y bodu `p1` je definována jako:

$$y = y_0 + 450 * a$$

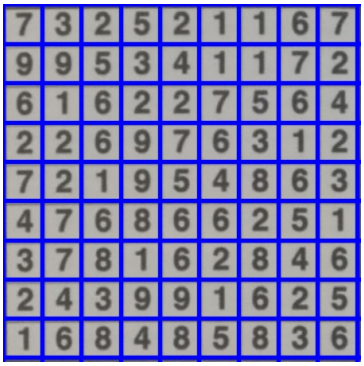
Souřadnice y bodu `p2` pak:

$$y = y_0 - 450 * a$$

kde:

- 450 je velikost obrázku,
- $a = \cos\theta$,
- $y_0 = b * \rho$, $b = \sin\theta$.

Poté jsou odstraněny duplicitní přímky a je určen výsledný počet vertikálních a horizontálních přímek. Podle počtu těchto přímek lze snadno odvodit velikost hracího pole. Ukázka je na obrázku 8.3.

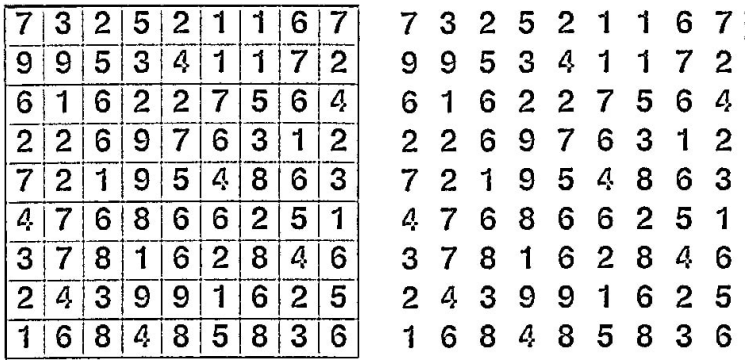


Obrázek 8.3: Nalezené přímky

8.1.6 Odstranění nežádoucích prvků

Za nežádoucí prvky jsou považovány přímky rozdělující hrací pole a šum. Nejprve je provedeno adaptivní prahování, které zajistí základní odstranění šumu. Potom je mřížka hracího pole překryta přímkami nalezenými v předchozím kroku. Na obrázku 8.4 je ukázka případu, kdy je tento postup dostačující.

Jak je ale vidět na obrázku 8.5, tak po pouhém překreslení přímek, jsou v obrázku stále pozůstalé fragmenty přímek rozdělujících hrací pole. Z tohoto důvodu jsou ještě aplikovány morfologické operace, viz sekce 3.4. Jsou použity dva strukturální elementy (jádra), jeden pro vertikální a druhý pro horizontální přímky. Obě jádra jsou obdélníky s různými velikostmi. Předpokládá se, že pozůstatky po překreslení nepřesahují velikost jedné buňky hracího pole. Tato velikost je již známa z předchozího kroku (je určena podle počtu dělících přímek). Velikosti obdélníkových jader jsou tedy `1` x `velikost buňky` pro odstranění horizontálních fragmentů a převrácená velikost pro vertikální. Pomocí těchto jader je provedena morfologická operace otevření.

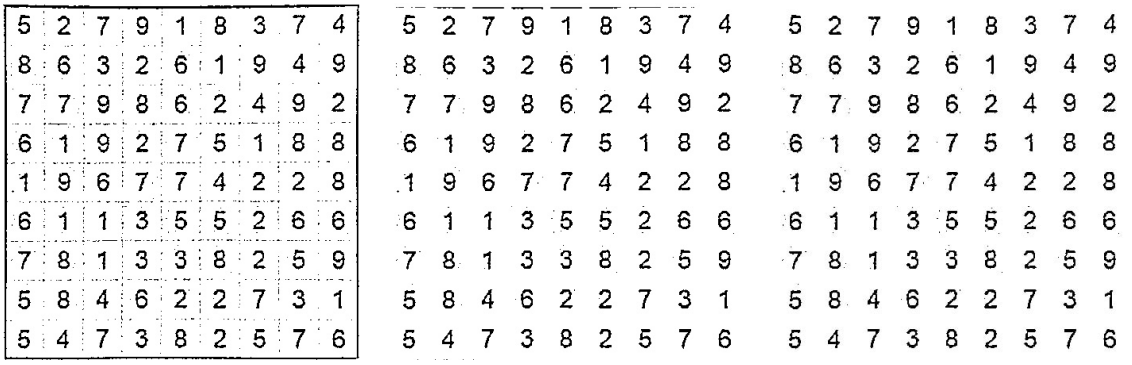


(a) Výsledek adaptivního prahování (b) Překreslené nalezené přímky

Obrázek 8.4: Odstranění mřížky hracího pole překreslením

8.1.7 Rozdělení na samostatné obrázky s čísly

Protože z předchozích kroků je již známá velikost hracího pole, nebrání nic tomu, aby byl obrázek s odstraněnými nežádoucími prvky rozdělen na samostatné buňky.



(a) Výsledek adaptivního prahování (b) Překreslené nalezené přímky (c) Odstranění fragmentů přímek pomocí morfologických operací

Obrázek 8.5: Odstranění mřížky hracího pole s využitím morfologických operací

Velikost každé buňky je určena jako podíl velikosti celého obrázku a velikosti hracího pole (`počet vertikálních přímek - 1 = počet horizontální přímek - 1`). V případě, že se počet vertikálních a horizontálních přímek neshoduje, je nastavena velikost 9, protože se jedná o nejčastější velikost hracího pole.

Vytvořené samostatné buňky jsou uloženy do kolekce `ArrayList` a následně jsou předány Tesseract pro další zpracování.

8.1.8 OCR

V tomto kroku probíhá samotné rozpoznávání číslic pomocí Tesseract (viz sekce 3.6). K tomu je implementována třída `TesseractOCR`, která instanciuje třídu `TessBaseAPI`.

Třída `TesseractOCR` ve svém konstruktoru kontroluje, jestli v zařízení existuje soubor s natrénovanými daty (jsou využita natrénovaná data dostupná na GitHubu[1]) a případně pro něj vytvoří cestu a nakopíruje jej do úložiště zařízení.

Poté inicializuje objekt třídy `TessBaseApi`, což zahrnuje připojení souboru s natrénovanými daty a nastavení jazyka pro rozpoznávání. Je použit anglický jazyk s omezením znaků na čísla 0-9.

Rozpoznávání znaků je prováděno v metodě `getResult()`, které je předán seznam bitmapových obrázků (každý obrázek nese jedno číslo). Na každý obrázek je volána funkce `getUTF8Text()`, která vrací rozpoznaný text jako řetězec. Metoda `meanConfidence()` vrací pravděpodobnost správnosti rozpoznávání jako integer v rozsahu 0-100.

Po rozpoznání každého jednoho čísla je vytvořen objekt `Cell` představující jednu buňku hracího pole. Po proiterování celého seznamu obrázků s čísly metoda `getResult` vrací dvourozměrné pole `Cell[][]` reprezentující výsledné hrací pole.

8.2 Hledání správného řešení

Řešení je nalezeno podle popisu v návrhu v kapitole 7 v částech 7.4 a 7.5. Pro hledání celkového řešení slouží třída `Solver` a pro hledání jednoho kroku třída `SolvingStepByStep`.

[1] https://github.com/tesseract-ocr/tessdata/blob/master/eng.traineddata

8.3 Grafické uživatelské rozhraní

Aplikace sestává ze sedmi aktivit. Jednou z nich je hlavní aktivita, která se dělí na dva fragmenty – fragment obsahující seznam uložených her a fragment umožňující vložení nové hry, kde je možné hru přidat buď prostřednictvím vyfotografování hracího pole, nebo výběrem fotografie z galerie.

Po výběru fotografie nebo vyfocení je uživatel přesunut do aktivity pro editaci fotografie. Potom následuje aktivita pro potvrzení rozpoznaného hracího pole, ze které se lze přepnout do aktivity pro editaci hracího pole. Další aktivitou je samotná hra. Jejich grafická podoba je k vidění na obrázku 9.2. Podrobněji jsou rozebrány níže.

Zbylé dvě aktivity jsou výběr fotografie z galerie a aktivita fotoaparátu. Kromě těchto dvou aktivit, jsou všechny ostatní aktivity dostupné pouze v orientaci na výšku.

8.3.1 Hlavní aktivita

Hlavní aktivitu tvoří dva fragmenty. Pro přepínání mezi fragmenty slouží spodní navigační lišta. Viz obrázek 8.6(a).

První z fragmentů zobrazuje hry uložené uživatelem. Vzhled připomíná galerii. Položky jsou seřazeny sestupně podle data vytvoření. Hlavním prvkem každé položky je náhledová fotografie (pořízena uživatelem). Nad náhledem je informace o tom, kdy byla hra vytvořena a pod náhledem se nachází čas strávený ve hře.

Původní návrh obsahoval ještě název hry, ale od toho bylo následně upuštěno, hlavním identifikátorem hry je tedy datum vytvoření, které by pravděpodobně stejně obsahoval automaticky vygenerovaný název hry.

Kliknutím na položku se otevře příslušná hra, kde se s malým zpožděním spustí časomíra. Po podržení položky lze danou položku smazat.

Druhý fragment slouží pro přidání nové hry. Uživatel má možnost vybrat fotografii z galerie nebo může fotografii rovnou pořídit. Podle toho se otevře buď aktivita fotoaparátu nebo aktivita výběru z galerie, kde uživatel může vybrat jen jednu položku.

Po získání fotografie hry je spuštěna aktivita pro editaci fotografie.

8.3.2 Aktivita pro editaci fotografie

V této aktivitě je uživatel vyzván k oříznutí fotografie na poměr stran 1:1, to je potřebné pro následné zpracování fotografie. Tato aktivita je spuštěna jak po výběru z galerie, tak po přímém vyfotografování hracího pole z aktivity fotoaparátu.

Aktivita nabízí dvě možnosti úpravy fotografie: oříznutí a rotaci. Mezi těmito možnostmi úpravy lze přepnout pomocí spodní lišty.

Oříznutí je možné provést pomocí gest, nebo s využitím lišty zobrazující procenta. Rotaci lze provést také pomocí gest nebo s využitím lišty zobrazující stupně. Pro rotaci je navíc k dispozici tlačítko pro otočení o 90° a tlačítko pro návrat k původní orientaci fotografie.

Po kliknutí na potvrzovací tlačítko v pravém horním rohu probíhá zpracování obrazu, tedy naskenování hracího pole. Toto tlačítko obsahuje animaci načítání. Snímek obrazovky je na obrázku 8.6(b).

8.3.3 Aktivita potvrzení rozpoznání

Aktivita obsahuje náhled fotografie pořízené uživatelem s již transformovanou perspektivou. Pod náhledem je rozpoznané hrací pole, kde se barevně zvýrazňují čísla, u kterých je menší

pravděpodobnost, že byla rozpoznána správně. Pomocí barev jsou označovány 4 stupně pravděpodobnosti správnosti rozpoznaní:

- čevená barva: 40% a méně,
- oranžová barva: 60%–41%,
- žlutá barva: 80%–61%,
- bílá barva: 81% a více.

Uživatel má možnost buď potvrdit rozpoznané hrací pole a spustit aktivitu hry, nebo přepnout na aktivitu editace, kde má možnost rozpoznané hrací pole upravit. K tomu slouží tlačítka v horní liště. Snímek obrazovky je na obrázku 8.6(c).

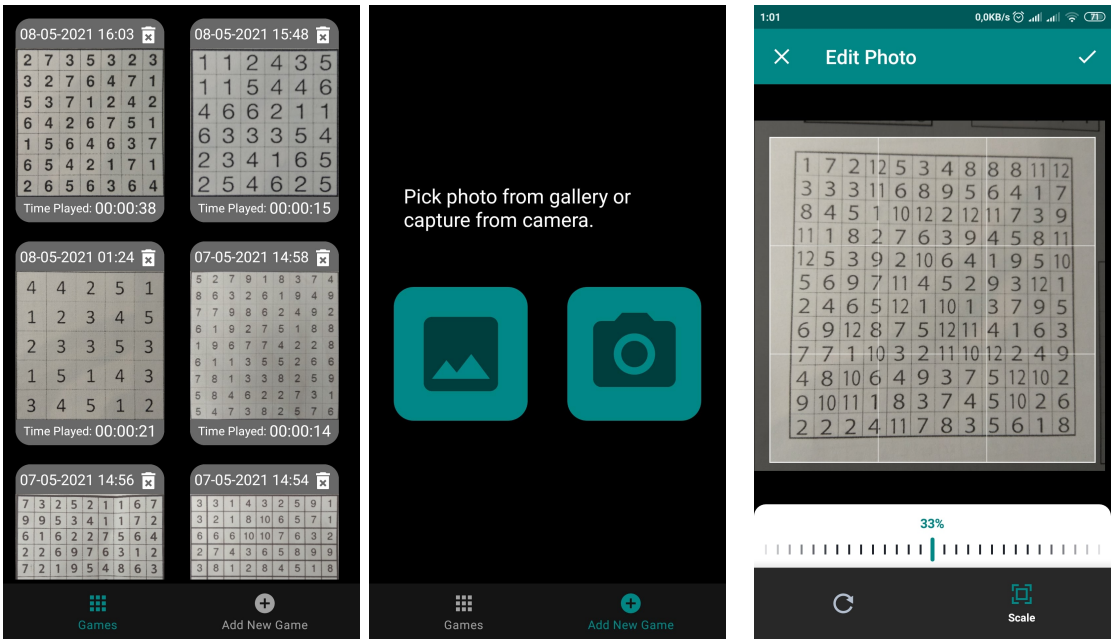
8.3.4 Aktivita editace hracího pole

V této aktivitě je uživateli umožněno opravit špatně rozpoznaná čísla, ale také může obarvit herní pole a začít tak rozehranou hru.

Uživatel má stále k dispozici náhled své fotografie pro porovnání. Pro úpravu je k dispozici virtuální klávesnice s číslicemi a tlačítky pro vymazání celé buňky nebo vymazání pouze poslední číslice.

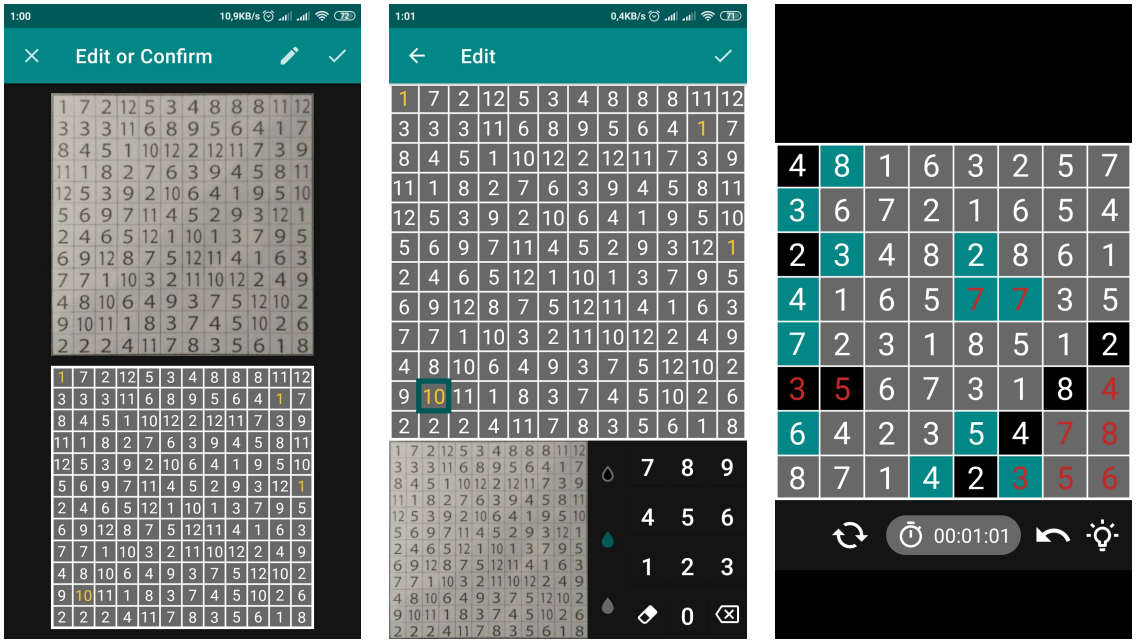
Před úpravou je potřeba nejdříve označit pole, které se má upravit. Po přepsání pole, které předtím bylo nějak barevně zvýrazněno, je číslo vykresleno bílou barvou (získává tak pravděpodobnost správnosti rozpoznání 100%).

Zobrazené hrací pole oproti tomu, jak je v aktivitě potvrzení rozpoznávání, využívá celou šířku obrazovky. To proto, že už nejde jen o náhled, ale o interaktivní prvek. U hracích polí větších rozměrů by mohlo být obtížné se na dotykovém displeji trefit do míněné buňky hracího pole. Viz obrázek 8.6(d).



(a) Hlavní aktivita sestávající ze dvou fragmentů

(b) Aktivita pro editaci fotografie



(c) Aktivita potvrzení rozpoznání

(d) Aktivita editace hracího pole

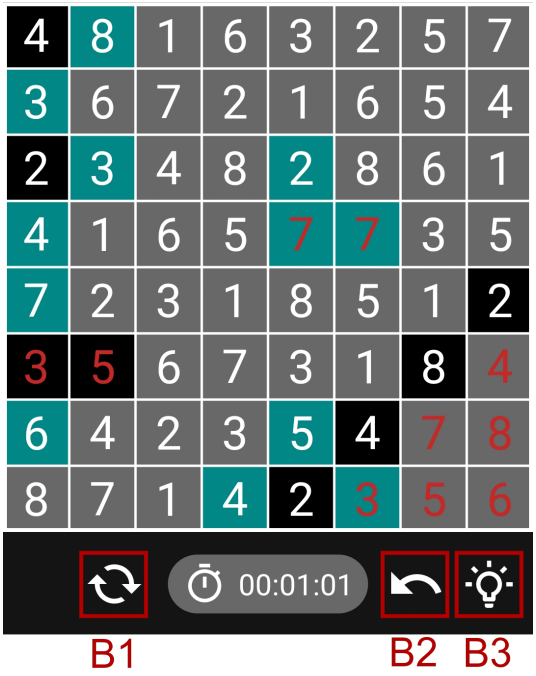
(e) Aktivita hry

Obrázek 8.6: Návrh grafického uživatelského rozhraní aplikace

8.3.5 Aktivita hry

Aplikace používá následovné značení políček: **černá barva** představuje smazané pole a **azurová barva** pole, které je potřeba ponechat. Pro zabarvení pole černou barvou je třeba na něj kliknout jednou, pro azurovou barvu dvakrát a pro odznačení (původní šedá) třikrát. Na obrázku 8.6(e) je snímek obrazovky, na kterém je vidět celkové rozložení této aktivity.

Na obrázku 8.7 je vidět, že jsou porušena všechna tři pravidla hry. To je uživateli naznačeno červeným zbarvením číslic.



Obrázek 8.7: Detail aktivity hry s popisem tlačítek

Mezi tlačítky uprostřed se nachází časomíra zaznamenávající čas strávený ve hře. Tlačítka podle značení v obrázku 8.7 mají funkcionalitu:

- tlačítko **B1** slouží pro restart hry, tzn. odbarvení všech políček a vynulování časomíry,
- tlačítko **B2** odstraní poslední krok (undo),
- tlačítko **B3** provede jeden krok za uživatele (nápověda).

Návrat do hlavní aktivity je možný systémovým tlačítkem zpět.

8.4 Databáze

Databáze je implementována pomocí knihovny Room, viz 5.4.1. Tvoří ji jedna tabulka představující hru. Hrací pole je do databáze ukládáno jako řetězec, kde jsou jednotlivé buňky oddělovány pomocí znaku „|“, hodnota buňky a stav buňky jsou oddělovány pomocným znakem „-“. V databázi se v tomto formátu kromě aktuálního stavu hracího pole při řešení ukládá i správné řešení dané hry.

Identifikační klíče her jsou generovány automaticky pomocí anotace `@PrimaryKey` s argumentem `autoGenerate = true`.

Kapitola 9

Testování

Výsledná aplikace je zveřejněna na Google Play (odkaz v QR kódu 9.1). To velmi usnadnilo celé testování tím, že mohlo probíhat na dálku. Tato kapitola popisuje jak testování probíhalo, co za chyby a nedostatky při něm bylo odhaleno a jaké úpravy byly v aplikaci provedeny.

Obrázek 9.1: QR kód nesoucí odkaz na produkční publikaci aplikace[1]

9.1 Testování publikace na Google Play

Aplikace byla zveřejněna na Goole Play v kanálu otevřeného testování. To znamená, že se testování mohl zúčastnit kdokoliv, kdo získal přístup přes odkaz. Nainstalováním aplikace z tohoto odkazu byl respondent automaticky registrován jako tester beta verze aplikace. Aby bylo možné získat přístup k pozdější produkční verzi, je potřeba opustit testování. To je možné provést z panelu aplikace na Google Play. Odregistrace Google účtu z testování nějakou dobu trvá.

Kvůli současné pandemické situaci bylo uživatelské testování znesnadněné tím, že nebylo možné se setkat se všemi respondenty osobně. Testování s mou přítomností tedy proběhlo jen s pěti respondenty z celkových deseti.

[1] https://play.google.com/store/apps/details?id=com.app.hitoriscanandplay

9.1.1 Průběh testování

Respondenti byli nejdříve dotázání, jestli znají hru Hitori a případně jim byla hra vysvětlena. Poté byl respondentům vysvětlen účel aplikace a byli seznámeni s průběhem testování. Následně byl testujícím zadán následující seznam úkolů:

1. Otevřete aplikaci a přejděte do části, kde je možné přidat novou hru.
2. Přidejte novou hru nahráním fotografie.
3. Zkontrolujte, zda byla čísla v hracím poli rozpoznána správně. Napoví Vám zbarvení číslic (žlutá, oranžová, červená).
4. Pokud nebyla některá čísla rozpoznána správně, editujte je.
5. Potvrďte a spusťte hru.
6. Pokuste se hru vyřešit (jestli Vám nejsou jasná pravidla hry nebo nevíte jak postupovat dál, využijte nápovědu).
7. Přidejte alespoň 5 Hitori.
8. Zobrazte uložené hry a vyberte jednu, kterou smažete.
9. Vyplňte dotazník (pokud Vám splnění některého z úkolů dělalo problém, zmiňte to v dotazníku pod otázkou 10. nebo 12.)

Pro vyhodnocení testování jsem použila online dotazník na Google Forms, který sloužil jako hlavní kanál pro zpětnou vazbu. Jeho podoba je k vidění v příloze A.

9.1.2 Výsledky testování

Celkové hodnocení aplikace bylo kladné a většina respondentů by aplikaci doporučila svým přátelům. Většina testujících si přála, aby aplikace zobrazovala správné řešení hry (to podle mě bylo způsobeno hlavně tím, že většina respondentů hru Hitori předtím neznala). Dvakrát se objevil návrh na přidání generátoru nových her. Často se taky objevovala zmínka, že je těžké poznat kdy je hra vyřešena.

U respondentů, se kterými jsem se mohla setkat a sledovat je při plnění úkolů, byly navíc vypozorovány další nedostatky jako:

1. Při prvním spuštění se uživatel dostane na obrazovku s uloženými hrami, kde ale ještě nic není.
2. Při ořezávání fotografie není uživateli jasné, že musí být vidět celý obrys hracího pole.
3. Při editaci rozpoznaného hracího pole uživatel očekává, že se po označení buňky a následném stisku číslice na klávesnici začne ihned přepisovat obsah pole. V případě kdy pole již obsahuje 2 znaky není pro další znak místo, a musí být nejdříve smazán předchozí obsah. Na to ale uživatel přijde docela rychle.
4. Uživateli není znám význam tlačítek obarvování hracích polí v editaci. Kvůli tomu někdy vytvoří hru, která nemá řešení a není možné ji spustit.

9.2 Řešení problémů a nedostatků odhalených při testování

V této části jsou popsány úpravy, které byly po testování provedeny. Nadpisy obsahují problémy a pod nimi je popsáno řešení daného problému.

Prázdný seznam her

Protože při prvním spuštění ještě nejsou uloženy žádné hry, bylo prohozeno pořadí fragmentu pro přidání nové hry a fragmentu obsahující seznam uložených her. Navíc se zobrazuje informace o tom že seznam je prázdný, protože tato situace může nastat nejen při prvním spuštění, ale i po smazání všech položek. Podoba fragmentu po úpravě je na obrázku 9.2(a).

Nejasnosti s ponecháním ohraničení hracího pole ve fotografii

Když uživatel ořízne fotografii tak, že v ní chybí ohraničení hracího pole, nedojde k nalezení hracího pole a je mu zobrazena hláška o neúspěšnosti skenování fotografie. Důvodů proč nedojde ke správnému skenování je více, uživatel je dopředu upozorněn, aby nechal všechny hranice viditelné. Znázorněno na obrázku 9.2(b).

Matoucí tlačítka

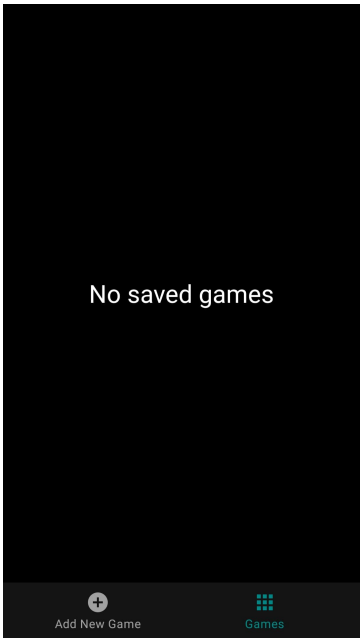
Jedná se o tlačítka pro obarvování buněk v aktivitě pro editaci rozpoznaného hracího pole. Kvůli těmto tlačítkům vnikaly hry, které neměly řešení a uživatel nevěděl, proč tomu tak je. Tlačítka byla z aktivity úplně odstraněna, díky tomu vzniklo i více místa pro číselnou klávesnici, viz obrázek 9.2(c).

Absence zobrazení správného řešení

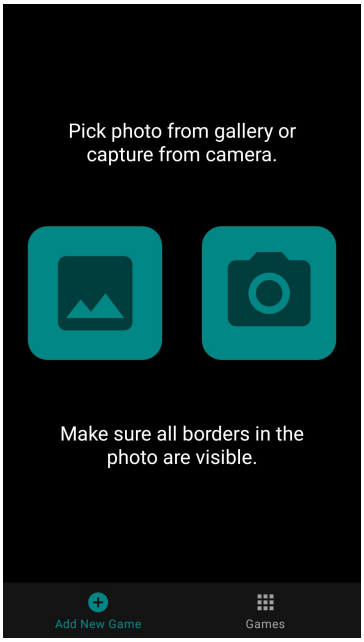
Většina respondentů v dotazníku uvedla, že by uvítali možnost zobrazení správného řešení hry. Bylo tedy přidáno tlačítko, jehož stlačení ukáže řešení. Zařazení tlačítka do GUI je zobrazeno na obrázku 9.2(d).

Dokončená hra je těžko rozpoznatelná

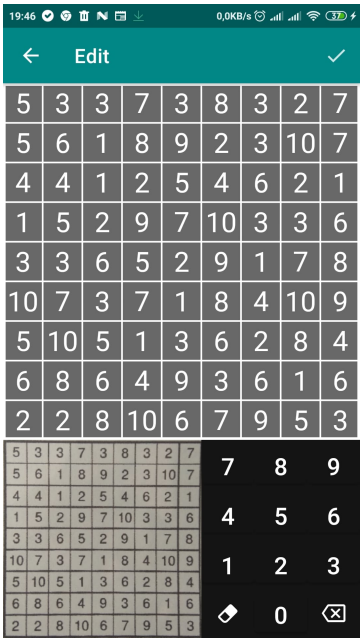
Ukázalo se že obarvení všech polí a zastavení časomíry není dostatečný indikátor dokončené hry. Proto byl po dokončení hry ještě přidán nápis „Congratulations“. Znázorněno na obrázku 9.2(e). Aby bylo možné rozeznat dokončenou hru v seznamu uložených her, je text „Time Played“ přepsán na „Finished in“. Viz obrázek 9.2(f).



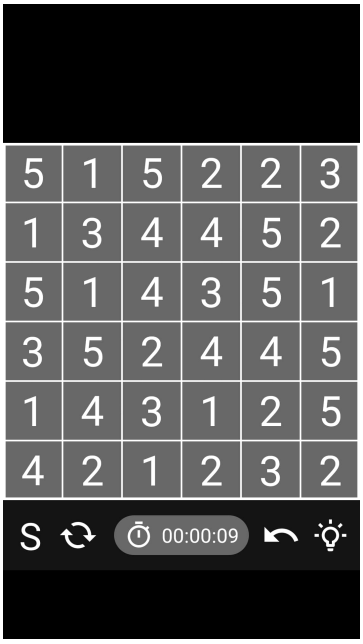
(a) Fragment s prázdným seznamem uložených her



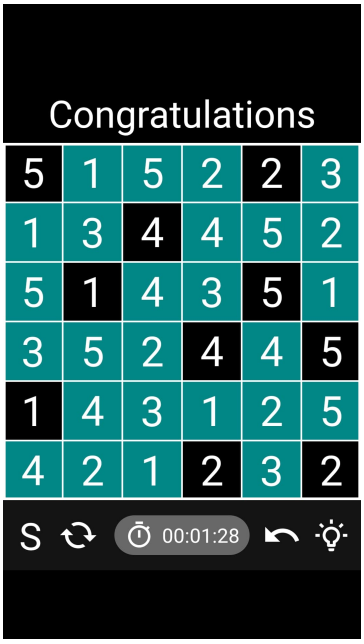
(b) Nové doporučení ve fragmentu pro přidání nové hry



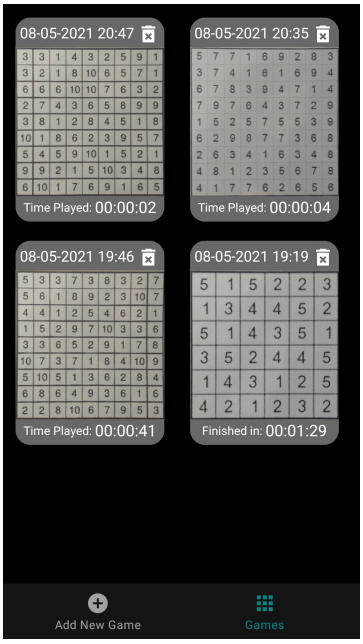
(c) Odstraněná matoucí tlačítka



(d) Přidané tlačítko pro zobrazení řešení



(e) Lepší znázornění dokončení hry



(f) Rozlišení dokončených her v seznamu

Obrázek 9.2: Snímky obrazovek znázorňující odstraněné nedostatky

Kapitola 10

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat Android aplikaci pro naskenování hry Hitori z novin a její dohrání. Pro dosažení tohoto cíle bylo třeba nastudovat techniky zpracování obrazu a problematiku návrhu a vývoje mobilních aplikací pro operační systém Android.

Aplikace byla otestována na vzorku uživatelů a byla zpracována zpětná vazba. Na základě zpětné vazby byla aplikace upravena. Výsledná aplikace je publikována na Google Play a může si ji kdokoliv stáhnout.

Na aplikaci plánuji dále pracovat a vylepšit ji na základě zpětné vazby z testování. Respondenti prostřednictvím zpětné vazby dávali najevo zájem o možnost generování nových her bez použití fotoaparátu. Tuto funkci bych v budoucnu ráda implementovala. Aplikace by se dala ještě vylepšit zdokonalením hledání správného řešení, které by mohlo být zrychleno. V neposlední řadě je třeba uvést přidání možnosti naskenování rozehrané hry. Zlepšení rozpoznávání číslic by bylo možné natrénováním vlastních dat omezených pouze na číslice.

Literatura

[1] *Codenames, Tags, and Build Numbers* [online]. [cit. 2021-01-12]. Dostupné z: https://source.android.com/setup/start/build-numbers#platform-code-names-versions-api-levels-and-ndk-releases.

[2] *Google kills Android distribution numbers on the web, but we've got you covered* [online]. [cit. 2021-01-13]. Dostupné z: https://9to5google.com/2020/04/10/google-kills-android-distribution-numbers-web/.

[3] *Hitori techniques* [online]. [cit. 2020-18-10]. Dostupné z: https://www.conceptispuzzles.com/index.aspx?uri=puzzle/hitori/techniques.

[4] *Hough Lines Transform Explained* [online]. [cit. 2021-04-20]. Dostupné z: https://medium.com/@tomasz.kacmajor/hough-lines-transform-explained-645feda072ab.

[5] *Mobile Operating System Market Share Worldwide* [online]. [cit. 2021-01-13]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[6] *Morphological Transformations* [online]. [cit. 2021-05-04]. Dostupné z: https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html.

[7] *OpenCV About* [online]. [cit. 2021-05-11]. Dostupné z: https://opencv.org/about/.

[8] *OpenCV Introduction* [online]. [cit. 2021-05-11]. Dostupné z: https://docs.opencv.org/3.4.11/d1/dfb/intro.html.

[9] *Platform Architecture* [online]. [cit. 2021-01-13]. Dostupné z: https://developer.android.com/guide/platform.

[10] *Save data in a local database using Room* [online]. [cit. 2021-05-02]. Dostupné z: https://developer.android.com/training/data-storage/room.

[11] *Understand the Activity Lifecycle* [online]. [cit. 2021-01-14]. Dostupné z: https://developer.android.com/guide/components/activities/activity-lifecycle.

[12] *Why you shouldn't rely entirely on Machine Learning* [online]. [cit. 2021-05-11]. Dostupné z: https://www.codementor.io/@alessandroflati/why-you-shouldn-t-rely-entirely-on-machine-learning-htgrbk1vu.

[13] Bradski, G. a Kaehler, A. *Learning OpenCV*. O'Reilly, 2008. 153-162 s. ISBN 978-0-596-51613-0.

[14] Canny, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE Computer Society. 1986, sv. 8, č. 6, s. 679–698. DOI: 10.1109/TPAMI.1986.4767851. ISSN 1939-3539.

[15] Gonzalez, R. C. a Woods, R. E. *Digital Image Processing*. 3. vyd. Pearson, 2008. 760-791 s. ISBN 978-0131687288.

[16] Kanopoulos, N., Vasanthavada, N. a Baker, R. L. Design of an Image Edge Detection Filter Using the Sobel Operator. *IEEE Journal of Solid-State Circuits*. 1988, sv. 23, č. 2, s. 358–367. DOI: 10.1109/4.996.

[17] Kubeš, P. Jak pokračuje vývoj umělé inteligence ve světě? [online]. 2020, [cit. 2021-05-09]. Dostupné z: https://elektro.tzb-info.cz/informacni-a-telekomunikacni-technologie/20355-jak-pokracuje-vyvoj-umele-inteligence-ve-svete.

[18] Kumar, V. Algorithms for Constraint-Satisfaction Problems: A Survey. *AI Magazine*. 1992, sv. 3, č. 1, s. 32. DOI: 10.1609/aimag.v13i1.976.

[19] Mardiris, V. a Chatzis, V. A Configurable Design for Morphological Erosion and Dilation Operations in Image Processing using Quantum-dot Cellular Automata. *Journal of Engineering Science and Technology Review*. 2016, sv. 9, s. 25–30. DOI: 10.25103/jestr.092.05.

[20] Mařík, V. *Umělá inteligence*. 1. vyd. Academia, 2013. ISBN 978-80-200-2276-9.

[21] McCorduck, P., Minsky, M., Selfridge, O. a Simon, H. A. History of Artificial Intelligence. In: *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1977, s. 951–954. IJCAI'77. DOI: 10.5555/1622943.1623034.

[22] Nixon, M. S. a Aguado, A. S. *Feature Extraction and Image Processing for Computer Vision*. 4. vyd. Academic Press, Inc., 2020. 103-106 s. ISBN 978-0-12-814976-8.

[23] Smith, R. W. History of the Tesseract OCR engine: what worked and what didn't. In: International Society of Optics and Photonics. *Document Recognition and Retrieval XX*. SPIE, 2013. DOI: 10.1117/12.2010051.

[24] Svitáková, L. Bystřílna: Hitori. *Mensa*. Mensa ČR. Únor 2017. ISSN 1211-8877.

Příloha A

Uživatelský dotazník

Dotazník k testování mobilní aplikace Hitori Scan & Play

*Povinné pole

1. Jak často používáte mobilní telefon? *

Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
Nepoužívám	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Používám denně

2. Znal/a jste předtím hru Hitori? *

Označte jen jednu elipsu.

- ◯ Ano
- ◯ Ne

3. Jak se Vám líbí design aplikace? *

Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
Nelíbí se mi	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Líbí se mi

4. Připadá Vám uživatelské rozhraní intuitivní? (ovládání je přehledné a jednoduché) *

Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
Špatné	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Výborné

5. Došlo ke správnému rozpoznání hracího pole? *

Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
Nikdy	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Vždycky

6. Využil/a jste možnost editace hracího pole? *

Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
Vůbec	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Vždycky

7. Uvítal/a byste možnost zobrazení správného řešení Hitori? *

Označte jen jednu elipsu.

- ◯ Ano
- ◯ Ne, Hitori si chci vyřešit sám/sama.

8. Jsou nějaké funkce, které v aplikaci nejsou, a uvítal/a byste je? *

Označte jen jednu elipsu.

- ◯ Ne
- ◯ Jiné: _____

9. Jak byste aplikaci celkově ohodnotil/a? *

Označte jen jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
Špatné	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Výborné

10. Proč jste hodnotil/a právě takto? *

11. Doporučil/a byste aplikaci přátelům? *

Označte jen jednu elipsu.

- Ano
- Ne

12. Prostor pro připomínky (nedostatky, návrhy na zlepšení)