VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DIGITALIZACE RUČNĚ PSANÝCH ZÁZNAMŮ ŠACHOVÝCH HER

DIGITIZATION OF HANDWRITTEN CHESS GAME SHEETS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **KRIŠTOF ŠIŠKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL ŠPANĚL, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

149525

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Šiška Krištof**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Digitalizace ručně psaných záznamů šachových her**
Kategorie:	Zpracování obrazu
Akademický rok:	2022/23

Zadání:

1. Vyhledejte a prostudujte existující metody a nástroje pro vyhledávání a rozpoznávání ručně psaného textu v obraze.
2. Analyzujte problém digitalizace ručně psaných šachových partiářů a dostupná řešení.
3. Připravte testovací sadu snímků pro vlastní experimenty.
4. Vyberte vhodné metody a nástroje a navrhněte řešení, které bude schopno ze snímků šachových partiářů extrahovat informace o sekvenci tahů a ve vhodném formátu ji exportovat.
5. Experimentujte s vaší implementací a případně navrhněte vlastní metody, které by zlepšily přesnost rozpoznání tahů.
6. Vyhodnoťte dosažené výsledky na testovací sadě snímků a popište možnosti budoucího vývoje.
7. Vytvořte stručný plakát nebo video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Eicher O., et al. *Handwritten Chess Scoresheet Recognition Using a Convolutional BiLSTM Network.* Proceedings of the International Conference on Document Analysis and Recognition Workshops, 2021, https://dl.acm.org/doi/abs/10.1007/978-3-030-86198-8_18.

Při obhajobě semestrální části projektu je požadováno:

- První čtyři body zadání.

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

Vedoucí práce:	**Španěl Michal, Ing., Ph.D.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	3.11.2022

Abstrakt

Šach je jedna z najpopulárnejších stolných hier sveta. Denne sa odohrá obrovský počet šachových hier a záujem spoločnosti o túto kráľovskú hru stále rastie. Pri šachových partiách odohraných naživo vznikajú prepisy šachových hier na šachové záznamy, inak nazývane aj partiáre. Prepis týchto partiárov do digitálnej podoby je monotónna práca trvajúca dlhú dobu. Čas strávený prepisom sa pri tom exponenciálne zvyšuje, ak je písmo nečitateľné alebo hra obsahuje veľké množstvo ťahov. Táto práca je zameraná práve na problém prepisu šachových partiárov do digitálnej podoby a zníženie času ľudskej práce nad touto potrebnou, avšak v mnoha oblastiach nezáživnou prácou.

Abstract

Chess is one of the most popular board games in the world. An enormous amount of chess game are played daily and its popularity is still on the rise. When playing live chess games, transcripts of the chess matches are created as chess records, also known as chess score sheets. Transcribing these score sheets into digital format is a tedious and time-consuming task. The time spent on transcription increases exponentially if the handwriting is illegible or if the game contains a large number of moves. This work focuses on the problem of transcribing chess score sheets into digital format and reducing the amount of time spent by humans on this necessary but often tedious task in many areas.

Kľúčové slová

spracovanie obrazu, OCR, rozpoznanie písma, šach, šachový záznam, spracovanie tabuliek

Keywords

image processing, OCR, text recognition, chess, chess scoresheet, table processing

Citácia

ŠIŠKA, Krištof. *Digitalizace ručně psaných záznamů šachových her*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Španěl, Ph.D.

Digitalizace ručně psaných záznamů šachových her

Prehlásenie

Prehlasuejm, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Michala Španěla Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Krištof Šiška
9. mája 2023

Poďakovanie

Ďakujem pánovi Ing. Michalovi Španělovi, Ph.D., za odbornú pomoc a cenné rady počas vypracovaní mojej bakalárskej práce.

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Šach je jedna z najpopulárnejších stolných hier na svete. Každý deň sa odohrá obrovský počet šachových hier a záujem o túto kráľovskú hru stále rastie. Potvrdzujú to aj štatistiky chess.com, v súčasnosti najpoužívanejšieho kanálu na hranie šachu. Štatistiky uvádzajú, že počet zaregistrovaných hráčov na tejto platforme bol v roku 2022 viac ako 100 miliónov. V porovnaní s rokom 2021 je to nárast o približne 25 miliónov a s rokom 2020 o 50 miliónov [1].

V súčasnosti sa šach hráva v dvoch prostrediach : online a naživo. Pri živých šachoch, na turnajoch alebo ligových zápasoch, je povinnosť hráča robiť prepis hry na šachový záznam. Tento záznam sa nazýva aj šachový partiár. Zo šachových záznamov je jednoduché presne určiť odohranú hru a jej výsledok. V online prostredí je šachová partia automaticky zaznamenávaná a uložená šachovým programom, cez ktorý užívateľ hráva partie.

Pri väčších turnajoch sa však stretávame s problémom prepisu šachových partiárov do digitálnej podoby. Digitalizácia šachových partiárov je dôležitá najmä pre samotných hráčov, ktorý sú schopný potom v budúcnosti nielen si tieto partie prejsť, ale aj analyzovať so šachovými nástrojmi. Prepis majú na starosti najmä organizátori týchto podujatí a pri účasti 200 hráčov vznikne na jedno kolo 100 hier, ktoré je nutné prepísať do digitálnej podoby. Toto je však časovo náročný proces a prepis jednej hry zvyčajne trvá približne 10 minút, avšak tento čas sa exponenciálne zvyšuje, ak je písmo nečitateľné, hra obsahuje viacej ťahov, alebo sa v partiároch nachádza neplatný ťah.

Táto práca je zameraná práve na problém prepisu partiárov do digitálnej podoby a zníženie času ľudskej práce nad touto potrebnou, avšak veľmi často nezáživnou a repetitívnou prácou. Cieľom práce je vytvoriť program, ktorý bude schopný zo snímku šachového partiára správne určiť jednotlivé ťahy oboch hráčov a exportovať ich v štandardnom formáte šachových hier PGN (portable game notation).

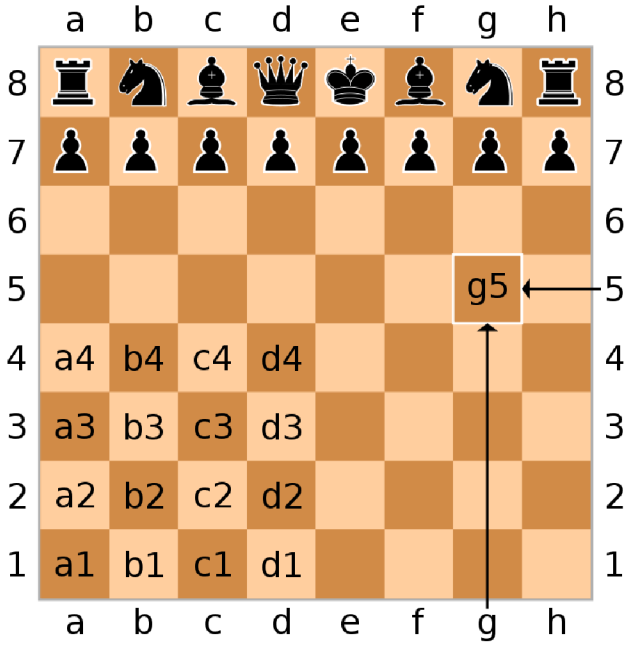
Výsledkom práce je systém, ktorý je schopný analyzovať vstupné snímky partiárov, segmentovať vstupný obraz na bunky ťahov a tieto bunky spracovať. Na spracovanie tabuliek a segmentáciu obrazu sa využili konvečné metódy spracovania obrazu založené na využití morfologických operácií a nájdení ohraničujúcich čiar. Pri využití nástroja `pytesseract` na rozpoznanie textu v jednotlivých bunkách dosahuje systém úspešnosť rozpoznania znakov 16,23 % pri partiároch najlepšej kvality.

Kapitola 2

Krátky úvod do šachu

Táto kapitola dáva čitateľovi nutné základné informácie na pochopenie riešenej problematiky. Úvod do PGN formátu, vysvetlenie jednotlivých ťahov, špeciálne ťahy a symboly, algebraická notácia.

Každá šachová partia pozostáva z ťahov čiernych a bielych figúrok, pričom ťahy sú robené na striedačku. Každý ťah na šachovnici je možné jednoznačne identifikovať. Na identifikáciu ťahov slúži algebraická notácia. Každé políčko na šachovnici je identifikovateľné unikátnymi koordinátmi – písmenom (a-h) a číslom (1-8). Horizontálne štvorce sa nazývajú rady a vertikálne štvorce stĺpce. Každý štvorec je teda reprezentaný písmenom a číslom, ako napríklad začiatočná pozícia bieleho kráľa je `e1`, čierneho kráľa `e8`.



Obr. 2.1: Políčka na šachovnici. Obrázok je prevzatý z [10].

Každý typ figúrky je identifikovaný jedným písemon. V slovenskom jazyku je to prvé písmeno názvu figúrky. Jednotlivé písmená v záznamoch majú teda nasledujúci význam podľa tabuľky 2.1.

K	Kráľ
D	Dáma
S	Strelec
J	Jazdec
V	Veža

Tabuľka 2.1: Význam znakov slovenskej šachovej notácie.

Ťahy sú identifikované písmenom pohnutej figúry a koordinátom políčka, na ktoré sa posunie. Pri ťahoch s pešiakmi sa identifikačné písmeno neudáva a je zaznačený len koordinát políčka, na ktoré sa má pešiak pohnúť.

Ak pohnutá figúra vyhodí počas svojho ťahu nepriateľskú figúru, táto skutočnosť sa môže značiť v notácií písmenom `x`.

Pokiaľ je možné na jedno políčko sa pohnúť rôznymi figúrami toho istého typu, je nutné pohnutú figúru identifikovať. Toto sa robí pridaním stĺpca, alebo radu, z ktorého pohnutá figúra vykonala ťah.

Promócia pešiaka po dosiahnutí poslednej súperovej rady sa značí pomocou znaku `'='`, po ktorom nasleduje typ figúry, na ktorú sa pešiak zmení.

V partiároch sa môžu vyskytovať aj niektoré špeciálne symboly. Ako napríklad `0-0` je symbol rošády kráľa s vežou na políčku h1 alebo h8. Symbol `0-0-0` značí rošádu kráľa s vežou na políčku a1 alebo a8. Podľa oficiálnych pravidiel FIDE však niektoré symboly hráči nie sú povinný značiť do partiáru a ich vynechanie nespôsobuje nevaliditu partiáru. Tieto symboly sú značenie brania `x`, šachu `+` a šach-matu `#` [11]. Ukážka a slovný popis šachovej notácie je uvedená v tabuľke 2.2.

Ťah	Význam ťahu
e4	Pešiak na e4
Jf3	Jazdec na f3
Dxg5	Dáma berie na g5
Jbxc4	Jazdec b berie na c4
e8=D	Pešiak na e8, dáma
g8=J	Pešiak na g8, jazdec
Kf1=	Kráľ na f1, ponúknutá remíza
Va8+	Veža na a8, šach
Dxf7#	Dáma berie na f7, šach mat

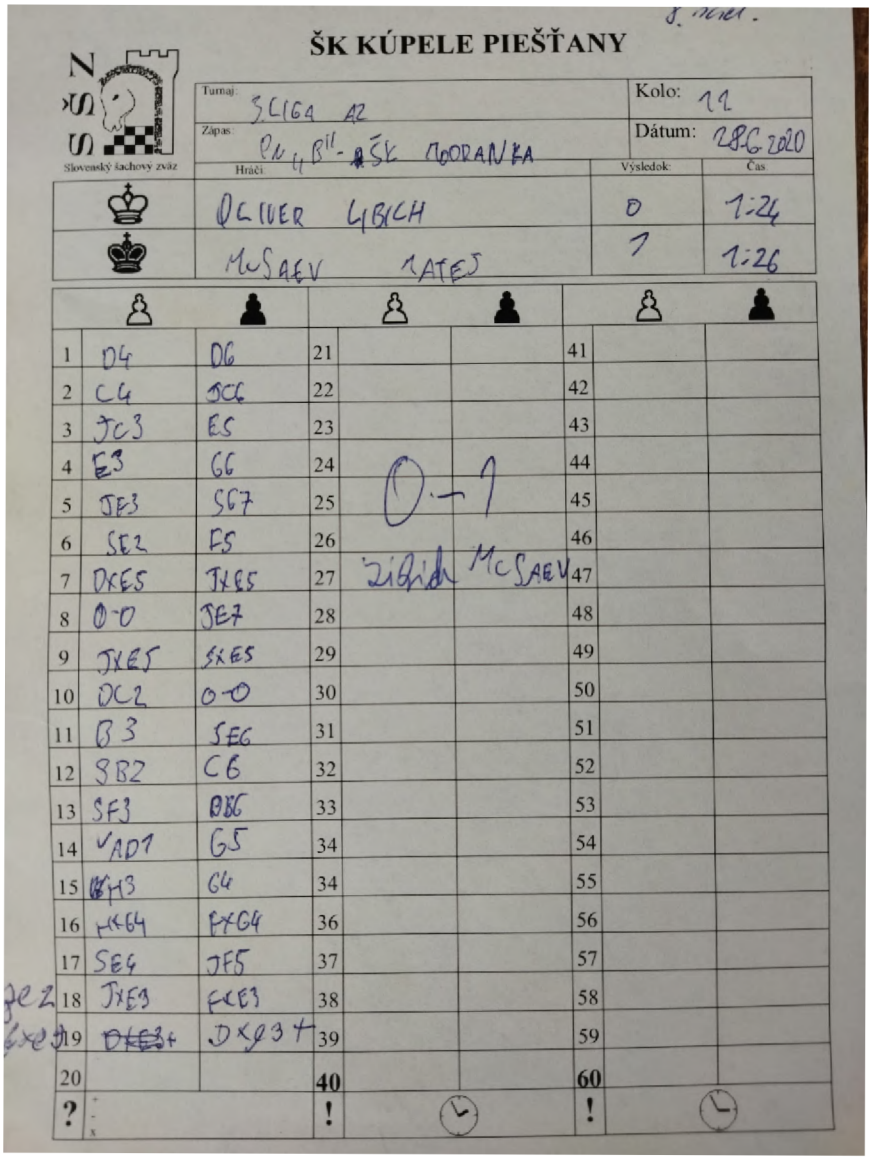
Tabuľka 2.2: Ťahy a ich vysvetlenie.

Koniec hry sa zapisuje pomocou čísiel `1`, `0` a `1/2`. 1 značí výhru, 0 prehru a `1/2` remízu. V samotnom zápise by teda značenie `1-0` znamenalo výhru pre bieleho, `0-1` výhru pre čierneho a `1/2-1/2` remízu.

2.1 Šachový partiár

Šachový partiár je papierový dokument slúžiaci na zaznamenanie ťahov jednej šachovej hry. Obvyklá veľkosť partiára je A5, pričom predná strana obsahuje priestor pre ťahy 1-60. Zadná strana obsahuje priestor pre ťahy nasledujúce po 60. ťahu. Predná strana obsahuje takisto aj hlavičku, do ktorej sú zapísané dôležité informácie ako mená hráčov, dátum hry, výsledok,

zápas, názov turnaja a podobné. Šachové partiáre nemajú jednotnú podobu a každý klub má vlastný grafický návrh, ktorý používa. Ukážku šachového partiáru je možné vidieť na obrázku 2.2.



Obr. 2.2: Šachový partiár. Cieľom systému je identifikovať jednotlivé bunky s ťahmi a previesť ich postupnosť do digitálneho šachového formátu.

2.2 PGN formát

Na uloženie hier do digitálneho prostredia sa používa PGN (portable game notation) formát. PGN je univerzálny formát pre šachové počítačové programy. Vychádza z anglickej šachovej algebraickej notácie. Je to obyčajný textový súbor, ktorý má špeciálnu príponu `.pgn`. Okrem uloženia samotnej hry, je možné k jednotlivým ťahom pridať aj komentáre. Tie sa dávajú to hranatých zátvoriek priamo za daný ťah. PGN formát takisto obsahuje aj hlavičku, ktorá obsahuje povinné a môže obsahovať aj voliteľné informácie :

`Povinné`

- Site – názov šachovej stránky (ak bola hra odohraná na internete).
- Event – názov turnaja, v ktorom bola hra odohratá.
- Date – dátum odohrania hry.
- Round – kolo.
- White – meno hráča za biele figúrky.
- Black – meno hráča za čierne figúrky.
- Result – výsledok hry.

`Voliteľné`

- ECO – identifikátor otvorenia.
- WhiteElo – elo hodnotenie hráča, hrajúceho za biele figúrky.
- BlackElo – elo hodnotenie hráča, hrajúceho za čierne figúrky.
- PlyCount – celkový počet odohraných ťahov.
- EventDate – dátum zahájenia turnaja, v ktorej bol odohraný zápas.

```
[Event "SVK 3.liga A2"]
[Site "?"]
[Date "2020.01.19"]
[Round "7.3"]
[White "Pericka, Tomas"]
[Black "Strapko, Tomas"]
[Result "1-0"]
[ECO "B31"]
[WhiteElo "1944"]
[BlackElo "1874"]
[PlyCount "99"]
[EventDate "2019.10.06"]

1. e4 c5 2. Nf3 Nc6 3. Bb5 g6 4. O-O Bg7 5. c3 Nf6 6. Re1 O-O 7. d4 cxd4 8.
cxd4 d6 9. h3 a6 10. Ba4 b5 11. Bb3 Na5 12. Bc2 Bb7 13. b3 Nc6 14. Nbd2 Rc8 15.
Bb2 Nb4 16. Bb1 a5 17. a3 Nc6 18. Bd3 b4 19. a4 e6 20. Qe2 Nh5 21. Ba6 Qb6 22.
Bxb7 Qxb7 23. Qb5 Qa7 24. Nc4 Nxd4 25. Bxd4 Bxd4 26. Nxd4 Qxd4 27. Rad1 Qc5 28.
Qxc5 Rxc5 29. Rxd6 Nf6 30. f3 Rfc8 31. Red1 Kf8 32. Rd8+ Ke7 33. Rxc8 Rxc8 34.
Nxa5 e5 35. Nc4 Nd7 36. Rxd7+ Kxd7 37. Nb6+ Kc7 38. Nxc8 Kxc8 39. Kf2 Kc7 40.
Ke3 Kd6 41. g3 g5 42. f4 f6 43. fxg5 fxg5 44. Kd3 Kc5 45. a5 Kb5 46. a6 Kxa6
47. Kc4 Ka5 48. Kd5 Kb5 49. Kxe5 Kc5 50. Kf6 1-0
```

Obr. 2.3: Ukážka PGN formátu.

Kapitola 3

Existujúce riešenia

V tejto kapitole sú rozobrané existujúce, voľne dostupné riešenia, zabývajuce sa problému digitalizácie šachových partiárov. Každé riešenie má popísané samotné fungovanie systému, jeho výhody, nevýhody a možnosti zavedenia do praxe.

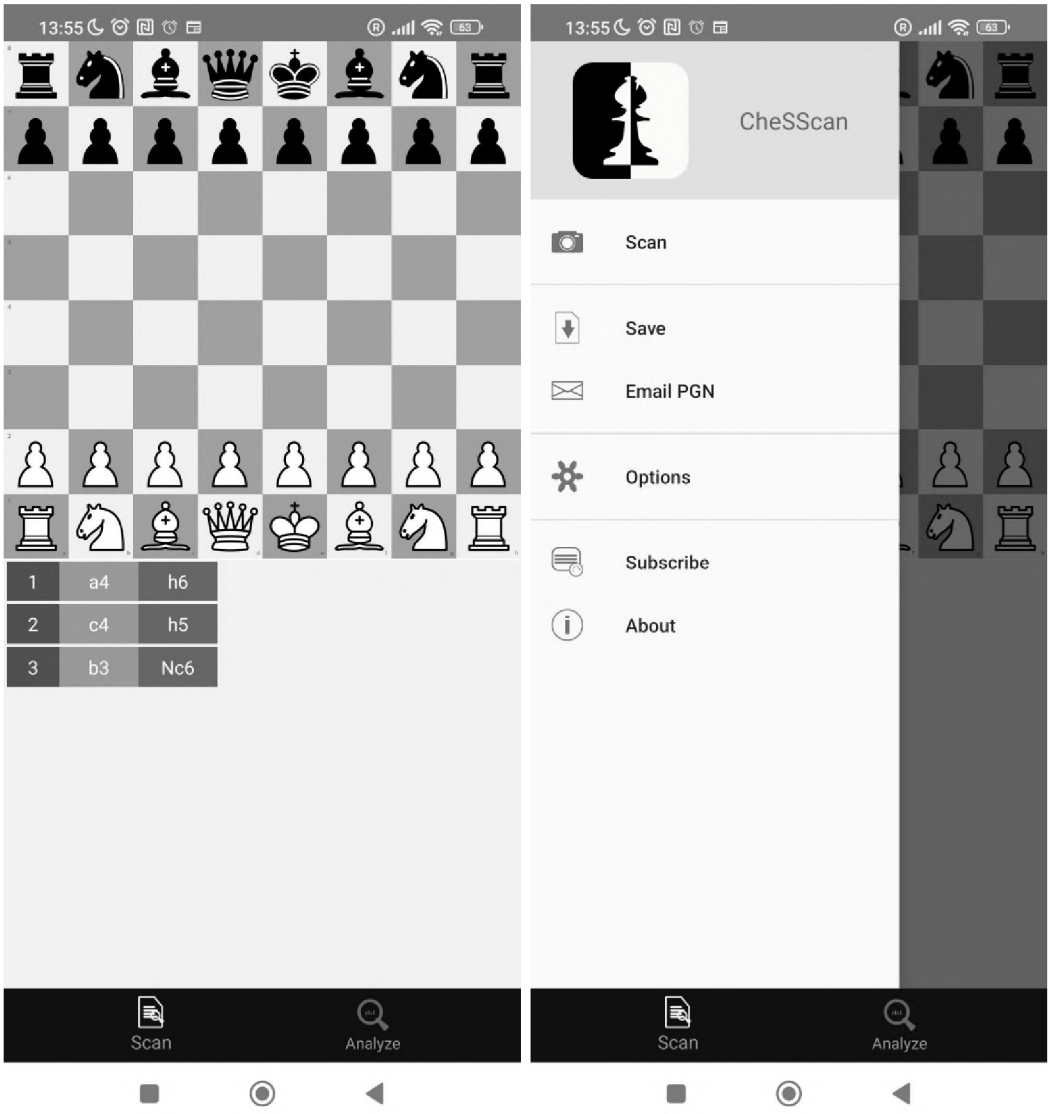
3.1 Chess Scan

Chess Scan je aplikácia pre mobilné zariadenia, dostupná cez google play. Užívateľ má urobiť snímku svojho partiára a systém z tejto fotky vytvorí PGN danej hry, ktorú si je možno rovno v aplikácií pozrieť. Má zabudovanú šachovnicu, kde je možné si pozrieť svoju súčasnú hru. Takisto ponúka možnosť danú hru odoslať na email, odkiaľ si ju užívateľ vie stiahnuť. Medzi výhody tejto aplikácie patrí aj možnosť upraviť ťahy, ktoré boli nesprávne vyhodnotené systémom, ako aj možnosť analyzovať hru pomocou zabudovaného šachového analyzátora.

V súčasnej verzií však nepodporuje slovenský jazyk v notácií a vygenerované výsledky s partiárov v slovenskom jazyku majú nulovú úspešnosť. Takisto je tu nemožnosť nahrať 2 partiáre tej istej hry, čo by bolo užitočné najmä pre organizátorov turnajov. Ďalšou nevýhodou je nemožnosť uložiť hru priamo na určité vzdialené úložisko, ako napríklad google drive. Ukážka uživateľského rozhrania je vidieť na obrázku 3.1.

3.2 PGN zo zaškrtávacieho zoznamu

Táto aplikácia využíva na značenie odohraných ťahov vlastný formát partiára. Každý ťah je teda postupnosť zaškrtnutých označení figúr a koordinátu políčka. Pre samotných hráčov je však táto možnosť veľmi limitujúca a to z viacerých dôvodov. Partiár, ktorý je nutný pre správne fungovanie programu má veľmi nezvyčajnú podobu, na ktorú hráči nie sú zvyknutý. Pred započatím hry by teda muselo byť účastníkom vysvetlené použitie partiára a je možné, že by počas hry hráči robili chyby v prepise, čo by mohlo viesť k časovým problémom. Ďalším problémom je veľkosť poskytnutého partiára. Celkovo poskytuje možnosť zápisu iba 32 ťahov. Priemerná dĺžka jednej šachovej hry je približne 40 ťahov [12], na čo tento druh partiáru nemá miesto. Ukážka partiáru je na obrázku 3.2. Z mojej vlastnej skúsenosti, ako ligového hráča, by nebolo možné využiť takýto druh partiáru, najmä z dôvodu absencie možnosti zaznačenia ťahu "brania"znakom **x**. Aj keď podľa pravidiel notácie, nie je nutné tento symbol používať, pre šachistov to má veľkú výhodu počas analyzovania hier po zápasoch, keďže tieto ťahy sú ľahko identifikovateľné a hrajúci hráč sa ľahšie zorientuje v sú-



Obr. 3.1: Aplikácia Chess Scan. Užívateľ je schopný pozrieť si vygenerovanú hru v zabudovanej šachovnici takisto aj opraviť zle vygenerované ťahy.

časnej pozícií. Táto analýza totiž prebieha naživo bezprostredne po odohraní partie. Keďže tento partiár bol navrhnutý za účelom uľahčenia rozpoznania písma počítačom, výsledkom pre ľudí je zlá čitateľnosť. Ukážka návrhu zaškrtávacieho zoznamu je vidieť na obrázku 3.2.

3.3 ReineChess

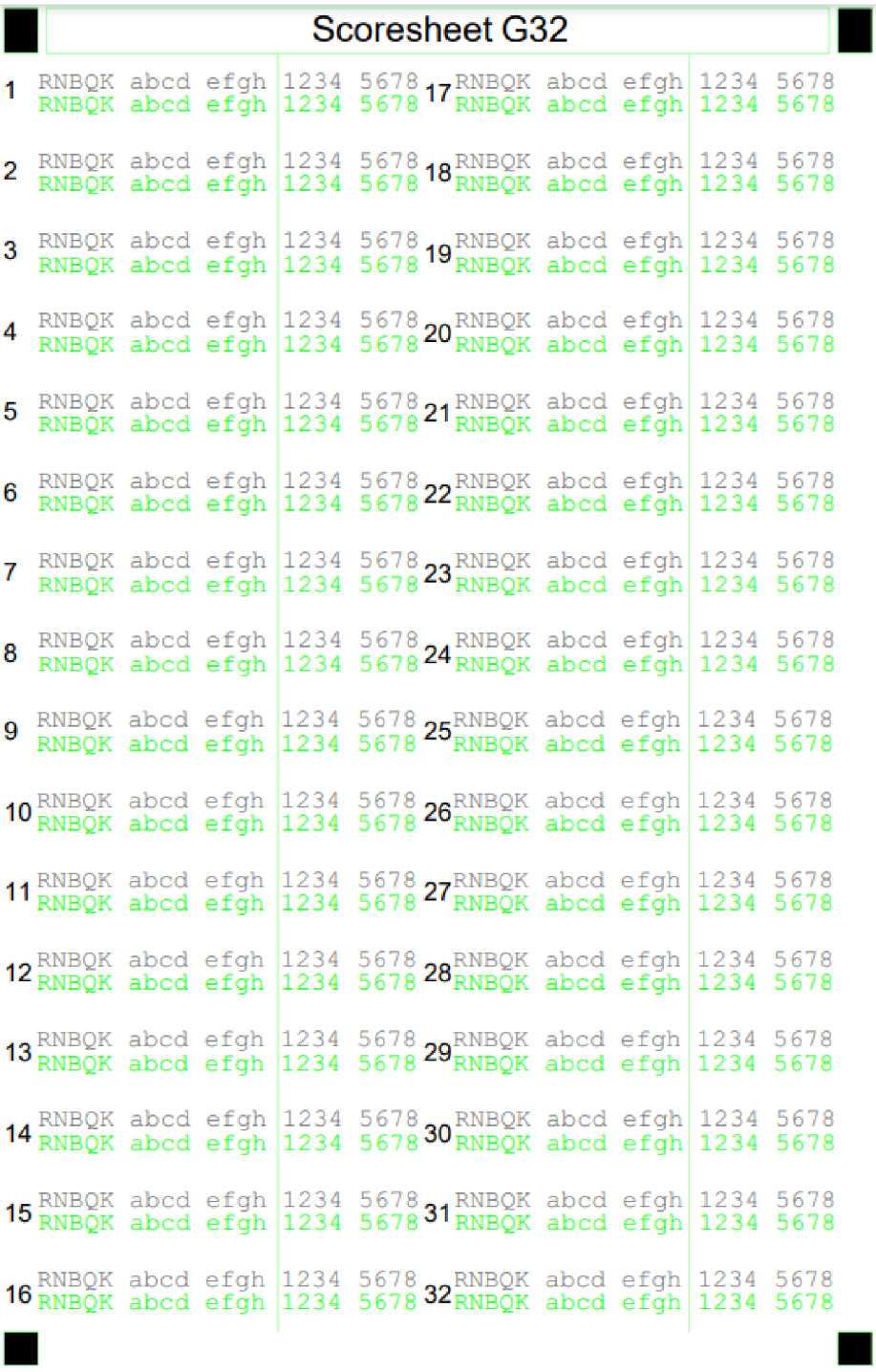
Je to webová aplikácia, ktorá využíva vlastný návrh šachového partiáru. Pre každý znak šachového ťahu je vytvorená bunka, do ktorej má byť daný ťah zapísaný. Aj keď je táto podoba partiáru nezvyčajná, pre šachistov by nebol veľmi veľký problém adaptovať sa na tento druh návrhu. Väčší problém by mohlo spôsobovať ťažké grafické odlíšenie medzi koncom buniek pre ťah bieleho a čierneho. Partiár poskytuje možnosť zápisu 50 ťahov, čo je pre priemernú dĺžku hry dostačujúci rozsah. Po nahratí partiáru by si mal byť hráč schopný uložiť vygenerovanú hru buď v `.pgn` súbore alebo ako `.txt` súbor. Takisto nie je užívateľom poskytnutá možnosť opravy zle vygenerovaných ťahov. Podľa autorov by mali hráči túto úpravu robiť prepisom textu v `.txt` súboroch. K odskúšaniu tohto riešenia však nedošlo, keďže stránka[1], na ktorej bol tento systém nasadený už v súčasnosti nie je dostupná. Návrh partiáru je možné vidieť na obrázku 3.3.

3.4 Riešenia v podobe výzkumných publikácií

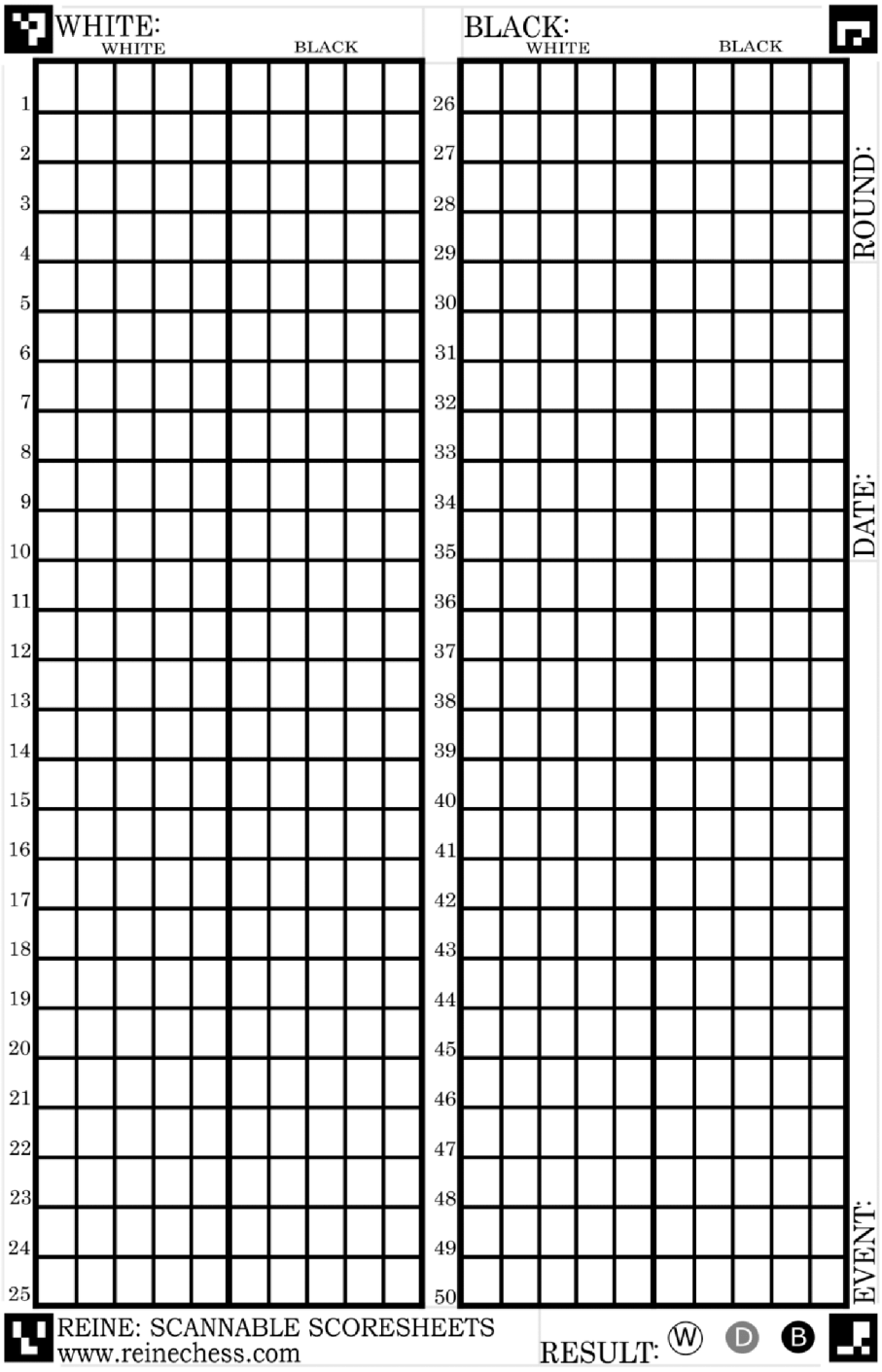
Niektoré riešenia sa nedajú otestovať, keďže nie sú dohľadateľné programové výstupy týchto riešení. Jedným takýmto príkladom je [13]. Autori tejto tejto práce využili na rozpoznanie písma obojsmernú rekurentnú neurónovú sieť (`BiLSTM`), ktorá bola trenóvaná na ich vlastnej dátovej sade, pozostávúcej z buniek šachových partiárov. Táto dátová sada[2], je však vytvorená z partiárov, ktoré sú písané v anglickej notácií. To znamená, že aj keď je tu mnoho podobností so slovenskou notáciou, názvy figúriek sú rozličné a teda niektoré znaky ako napríklad `J`, čo v slovenskej notácií značí jazdca, sa v tejto dátovej sade nevyskytujú ani raz a teda táto dátová sada nie je plno využiteľná pre slovenskú notáciu. Pre slovenskú notáciu neexistuje žiadna voľne dostupná dátová sada na problém digitalizácie šachových partiárov.

[1] `https://www.reinechess.com/`

[2] `https://sites.google.com/view/chess-scoresheet-dataset`



Obr. 3.2: Zaškrtávací zoznam.



Obr. 3.3: Partiár aplikácie reinechess.

Kapitola 4

Spracovanie obrazu pre rozpoznanie textu

Táto kapitola oboznamuje čitateľa s technikami spracovania obrazu, využívanými metódami a výberu OCR systému pre problematiku digitalizácie šachových partiárov.

4.1 Získavanie snímku

Prvou časťou rozpoznania textu v obraze je obstaranie samotného snímku dokumentu. V súčasnosti sa využívajú 2 hlavné prístupy na získanie týchto snímkov a to za použitia digitálneho fotoaparátu alebo skenom daného dokumentu. Výsledok skenovania zvyčajne poskytuje lepšie, presnejšie výsledky, avšak použitie fotoaparátu je rýchlejšie, dostupnejšie a jednoduchšie.

Z matematického pohľadu je snímok dvojrozmerná matica hodnôt intenzity p_{ij}, ktoré sa obvykle nazývajú pixely [14]. Pixel alebo pixelový element je najmenší adresovateľný prvok zobrazovacieho zaradenia (displeju). Pixel má štvorcový tvar, s hranami o dĺžke 1, s hodnotou p_{ij} v bode (i, j). Podľa toho, aké hodnoty môžu pixely nabývať, delíme snímky do 3 kategórií, ukážka týchto typov je vidieť na obrázku 4.1 :

- **Čierno-biele** – Toto je najjednoduchšia forma snímkov, kde jednotlivé hodnoty pixelov p_{ij} môžu naberať jednu z hodnôt z dvojice `{0,1}`. 0 reprezentuje čiernu farbu a 1 bielu farbu.
- **Šedotónové** – Hodnota p_{ij} je celé číslo z rozmedzia hodnôt 0-255, ktoré reprezentujú odtieň šedej farby.
- **Farebné** – Každá hodnota p_{ij} je vektor troch hodnôt. Pokiaľ je snímok zaznamenaný v RGB modely, každý $p_{ij} = (r_{ij}, g_{ij}, b_{ij})$, ktoré definujú množstvo červenej, zelenej a modrej farby v bode (i, j)

Obr. 4.1: Rôzne typy snímkov. Pôvodný farebný obrázok je prevzatý z [15].

4.2 Predspracovanie

Hlavným cieľom predspracovania v OCR systémoch je zlepšiť kvalitu obrazu dokumentu tak, aby bolo možné správne rozpoznať a interpretovať na ňom obsiahnutý text. Proces predspracovania zahŕňa rôzne kroky, ako je napríklad odstraňovanie šumu, vyrovnávanie kontrastu a normalizácia osvetlenia, ktoré môžu pomôcť rozpoznávaču pri rozpoznávaní znakov. V tejto sekcií sú popísané iba metódy, ktoré sú relevantné k procesu digitalizácie partiárov.

Normalizácia

Normalizácia sa používa na konzistentné spracovanie dokumentov rôznych veľkostí a formátov. Cieľom normalizácie je upraviť rozmer dokumentu, rozloženie textu a veľkosť písma, aby sa zabezpečila konzistentnosť a jednotnosť výsledkov OCR.

Binarizácia

Binarizácia je prevod obrazu na jeho čierno-bielu podobu [4.1]. Na tento proces sú v súčasnosti najpoužívanejšie nasledujúce techniky :

- **Globálne prahovanie** – Najjednoduchšia technika. Pre celý obrázok sa vyberie hraničná hodnota, podľa ktorej sa pre jednotlivé pixely určí porovnaním ich hodnota. Pre hraničnú hodnotu X teda platí :

$$p_{x,y} = \begin{cases} 1 & \text{Ak } p_{x,y} >= X \\ 0 & \text{Inak} \end{cases} \tag{4.1}$$

  Kde x, y sú súradnice pixelu snímku, $p_{x,y}$ je farba pixelu v bode x, y a X je vypočítaná hodnota prahu.

- **Adaptívne prahovanie** – Pre každý pixel je vypočítaná hraničná hodnota, podľa jeho susediacich pixelov

- **Otsu binarizácia** – Automaticky vytvorí optimálny prah, na základe histogramu [16].

- **Sauvola prahovanie** – Používa sa najmä v prípadoch, kedy pozadie textu nie je jednotné napríklad z dôvodu osvetlenia. Nevyužíva jednu globálnu hodnotu prahu, ale pre každý pixel je vypočítaných niekoľko prahov, ktoré berú do úvahy priemer a štandardnú odchýlku miestneho okolia [17].

4.3 Dilatácia

Je to morfologická operácia, ktorá sa používa prevažne na zvýraznenie hraníc a zväčšenie objektov. Vo svojej podstate dilatácia zväčšuje oblasť svetla a zmenšuje oblasť tmy. V jednoduchom ponímaní je dilatácia proces, pri ktorom sa určí matica nazývaná šablóna, ktorá je potom aplikovaná na každý pixel v obraze a pokiaľ sú niektoré pixely svetlejšie ako aktuálny pixel, tento pixel je označený ako svetlý. Podrobnejšie vysvetlenie je možné nájsť tu [2].

4.4 Erózia

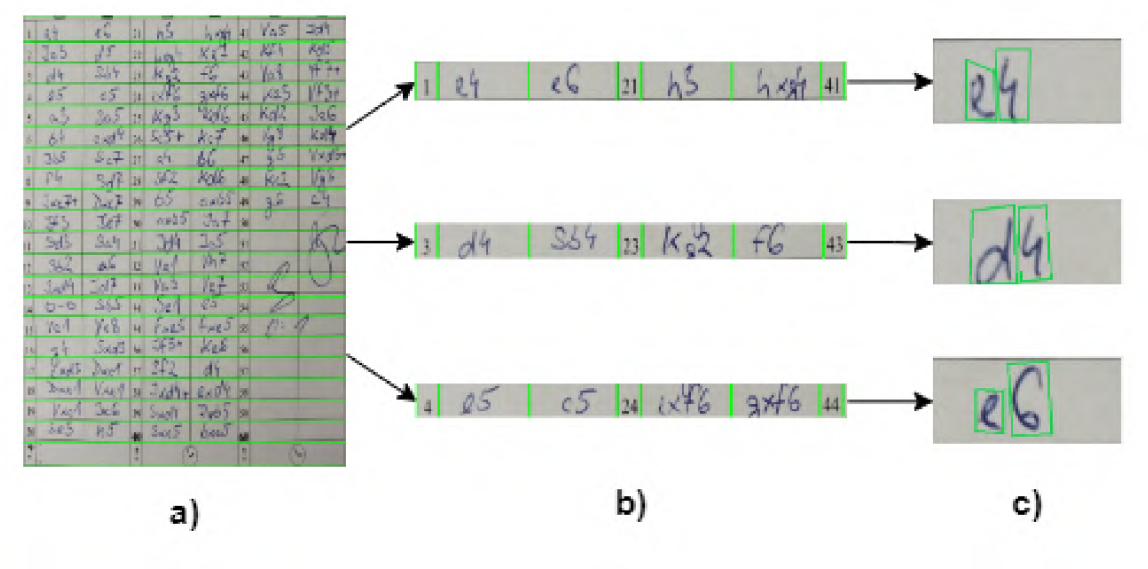
Je to morfoligcká operácia, používaná na odstránenie malých objektov, vyhladenie hrán a oddelenie spojených objektov. Na rozdiel od dilatácie, zväčšuje oblasť tmy a zmenšuje oblasť svetla. Jej princíp fungovania je podobný dilatácií, teda pomocou matice aplikovanej na každý pixel v obraze. Ak sú niektoré pixely v okolí aktuálneho pixelu tmavšie ako daná hodnota, tak sa tento pixel označí ako tmavý. Podrobnejšie vysvetlenie je možné nájsť tu [2]

4.5 Segmentácia

Táto sekcia je založená na [3]. Segmentácia textu je prevádzaná na 3 úrovniach. Výber segmentačnej úrovne textu závisí od konkrétnej aplikácie a jej potrieb. Ukážka rôznych úrovní segmentácie je vidieť na obrázku 4.2.

- **Segmentácia na úrovni riadkov** – prvá úroveň segmentácie. Cieľom je zo vstupného obrazu extrahovať jednotlivé riadky textu.
- **Segmentácia na úrovni slov** – druhá úroveň segmentácie. Cieľom je rozdeliť vstupný riadok na jednotlivé slová. V čisto textových dokumentoch sa môže využiť skutočnosť, že medzery medzi jednotlivými slovami sú často väčšie než medzery medzi znakmi v slove.
- **Segmentácia na úrovni znakov** – tretia úroveň segmentácie. Cieľom je rozdeliť slová na jednotlivé znaky.

V kontexte digitalizácie šachových partiárov, kde jednotlivé ťahy sú rozdelené do buniek, dáva najväčší zmysel segmentácia na druhej úrovni. Avšak oproti segmentácie čisto textových dokumentov je možné využiť faktu, že jednotlivé slová sa nachádzajú v predom definovaných bunkách tabuľky. Narozdiel od segmentácie riadkov a hľadaní väčších medzier je teda možné sa zamerať na rozpoznanie buniek tabuľky, čím sa dá dostať na druhý segmentačný level.



Obr. 4.2: Ukážka úrovní segmentácie. **a)** Prvá úroveň. Vstup je segmentovaný na riadky. **b)** Druhá úroveň. Riadok je segmentovaný na slová. **c)** Tretia úroveň. Slová sú segmentované na jednotlivé znaky.

4.6 Rozpoznanie písma

OCR (optical character recognition) je proces, ktorého cieľom je konvertovať snímok písma do počítačovo čitateľného textového formátu. OCR systémy majú množstvo kategórií v ktorých sa líšia. Na základe typu písma sa delia do dvoch kategórií.

- OCR systémy pre strojom tlačené písmo – veľké množstvo fontov
- OCR systémy pre rukou písané písmo – individuálny štýl písania. Každý človek má unikátny štýl písma.

Zvyčajne sú systémy pre strojom tlačené písmo spoľahlivejšie a presnejšie. Hlavnou príčinou je fakt, že tlačené znaky sú štandardizované a ľahšie od seba odlíšiteľné. Takisto má menej problémov ako systémy pre rukou písaný text, ktoré musia zohľadňovať pri svojej práci širšiu škálu variácií a typov písma a samotnú čitateľnosť textu.

Tesseract

Tesseract[1] je systém na rozpoznanie písma, pôvodne vyvíjaný spoločnosťou Hewlett Packard v rokoch 1984-1994. Od roku 2005 bol systém vydaný ako open source a v súčasnosti je udržiavaný spoločnosťou Google. Má v sebe zabudovanú podporu viac ako 100 jazykov. Užívateľom poskytuje možnosť trénovania na vlastných dátových sadách. Systém je takisto schopný automatickej segmentácie vstupného obrazu.

[1] `https://github.com/tesseract-ocr/tesseract`

4.7 Postprocessing

Cieľom postprocessingu je nájsť a opraviť chyby vo vygenerovanom texte. V kontexte digitalizácie partiárov sú významné najmä tieto metódy :

- **Manúalna korekcia** – Chyby vo vygenerovanom texte sú opravené človekom.
- **Oprava chýb založená na zozname možných slov** – Táto metóda porovnáva vygenerovaný text s uloženými vstupmi v zozname možných slov a vypočítava ich vzdialenosť. Vzdialenosť je počet krokov (inzercia, odstránenie a zmena), vďaka ktorým sa dokáže transformovať text **A** na text **B**. Príkladom tejto metódy je napríklad Levenshteinova vzdialenosť [4].

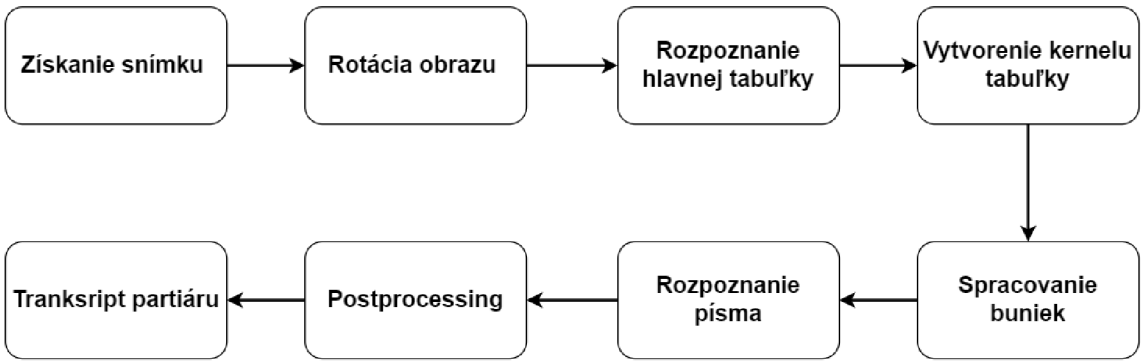
Kapitola 5

Návrh systému

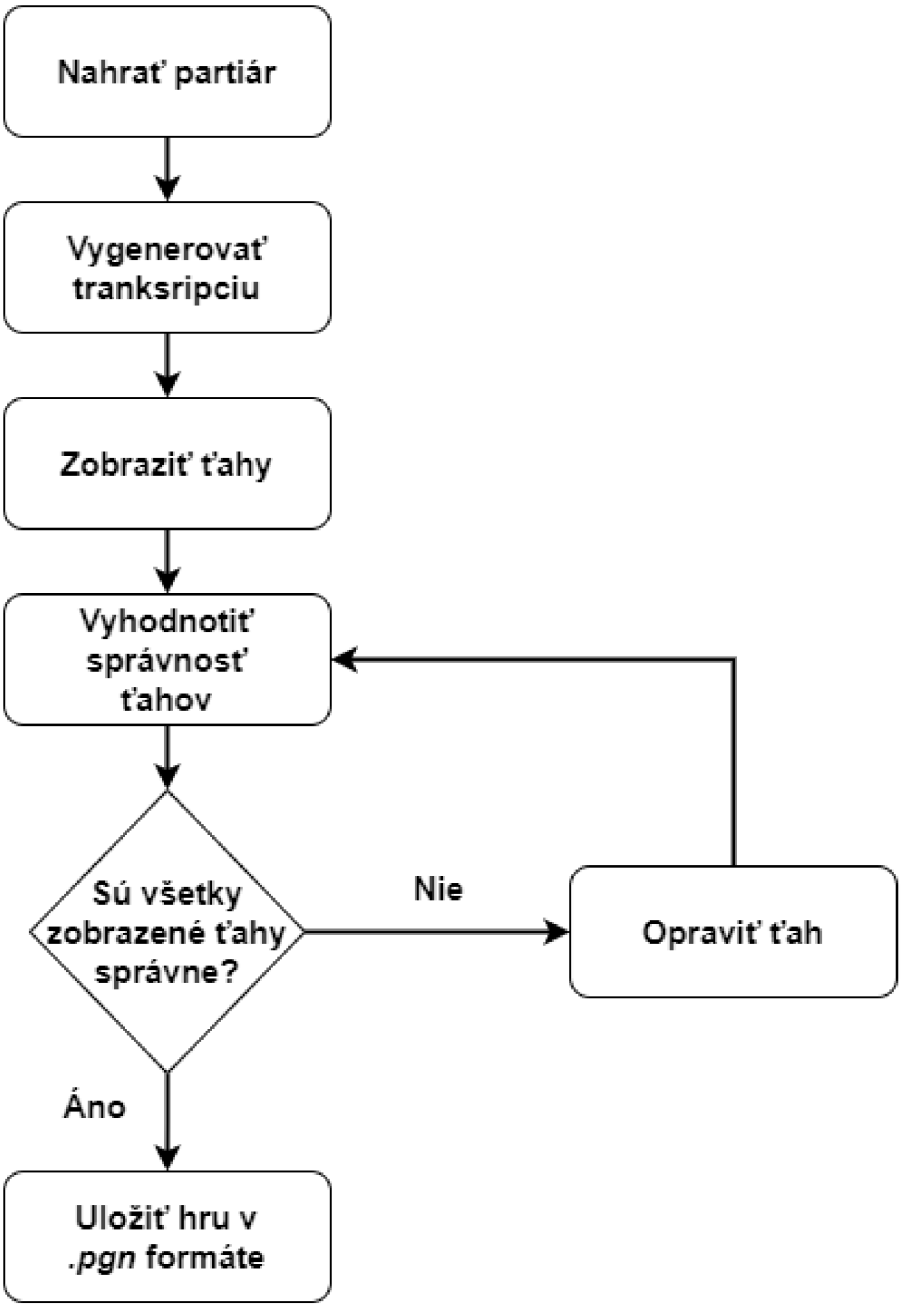
Táto kapitola popisuje navrhnutý systém, jeho fungovanie, postupnosť jednotlivých krokov práce s partiárom a vysvetlenie výberu použitia konkrétnych prístupov.

Aplikácia by mala umožniť užívateľovi nahratie snímku, automatickú segmentáciu buniek na ťahy, vygenerovanie prepisu jednotlivých buniek na text, možnosť opravy zle vygenerovaných ťahov a možnosť uložiť opravený prepis vo formáte `pgn`. Postup práce užívateľa so systémom je možné vidieť na obrázku 5.2.

Jednou z funkcií výsledného systému je rozpoznanie jednotlivých buniek s ťahmi a ich spracovanie. Výsledok predspracovania buniek tabuľky by mali byť binárne obrázky s ťahom, ktoré obsahujú minimálny počet rušivých elementov. Za rušivé elementy sa považujú hlavne zbytky čiar ohraničujúcich bunku a iné malé objekty, ktoré vznikli počas predspracovania obrazu. Diagram postupu práce systému je možné vidieť na obrázku 5.1.



Obr. 5.1: Diagram postupu práce systému pri vytváraní tranksriptu partiáru šachovej hry.



Obr. 5.2: Diagram práce užívateľa so systémom.

5.1 Získanie snímku

Systém je schopný pracovať zo snímkami získanými pomocou digitálneho fotoaparátu alebo skeneru. V prípade fotoaparátu je nutné počítať s problémami ako je zrotovanie snímku a samotného textu a skreslenie obrazu s nedostatočným rozlíšením ťahov.

Výstup systému priamo závisí od kvality dodaného snímku. Pokiaľ sú vstupné snímky nekvalitné, je možné očakávať zníženú presnosť rozpoznania ťahov. Pre čo najlepšie výsledky je nutné dodať kvalitný snímok, s dostatočným rozlíšením, celým hracím partiárom

a minimálnym zachytením podložky, na ktorom bol partiár fotení (v prípade použitia fotoaparátu).

5.2 Rotácia obrazu

Prvou časťou predspracovania je vyriešenie problému spojeným so získavaním snímku – zrotovaný obraz. Nerovné čiary čiary a text by spôsobilo nepresnejšiu segmentáciu jednotlivých buniek s ťahmi a zmenšilo by šance na úspešný prevod obrazu na text pomocou zvoleného OCR systému. Na rotáciu bol vytvorený algoritmus, ktorý pracuje v nasledujúcich bodoch :

1. Získanie obrazových čiar pomocou Houghovej transformácie.
2. Vyfiltrovanie dostatočne dlhých, horizontálnych čiar.
3. Zistenie priemerného sklonu daných horizontálnych čiar.
4. Vypočítanie uhla rotácie z priemerného sklonu.
5. Vyrovnanie obrazu pomocou zisteného uhla rotácie.

5.3 Rozpoznanie hlavnej tabuľky

Samotný partiár obsahuje hlavičku, ktorej spracovanie však táto práca nerieši. Toto rozhodnutie je najmä kvôli tomu, že pre hráča nie je potrebné pre analýzu hier poznať informácie ako názov turnaja, kolo, dátum a podobné. Pre uskladnenie vytvorených hier však tieto informácie potrebné sú. Po vygenerovaní a otvorení hry v náhodnom šachovom programe napríklad lichess[1] môže hráč doplniť chýbajúce informácie priamo do `pgn` súboru.

Vzhľadom na dobrú definíciu úlohy a predpokladaných vstupných snímkov som sa rozhodol využiť konvenčné metódy pre spracovanie obrazu. Alternatívnym prístupom je napríklad použitie architektúr založených na strojovom učení za použitia architektúr na detekciu objektov. Príkladom takéhoto modelu je napríklad architektúra R-CNN [5]. Pri použití konvenčných metód som sa takisto mohol vyhnúť problémom s obstarávaním dátovej sady na trénovanie týchto modelov.

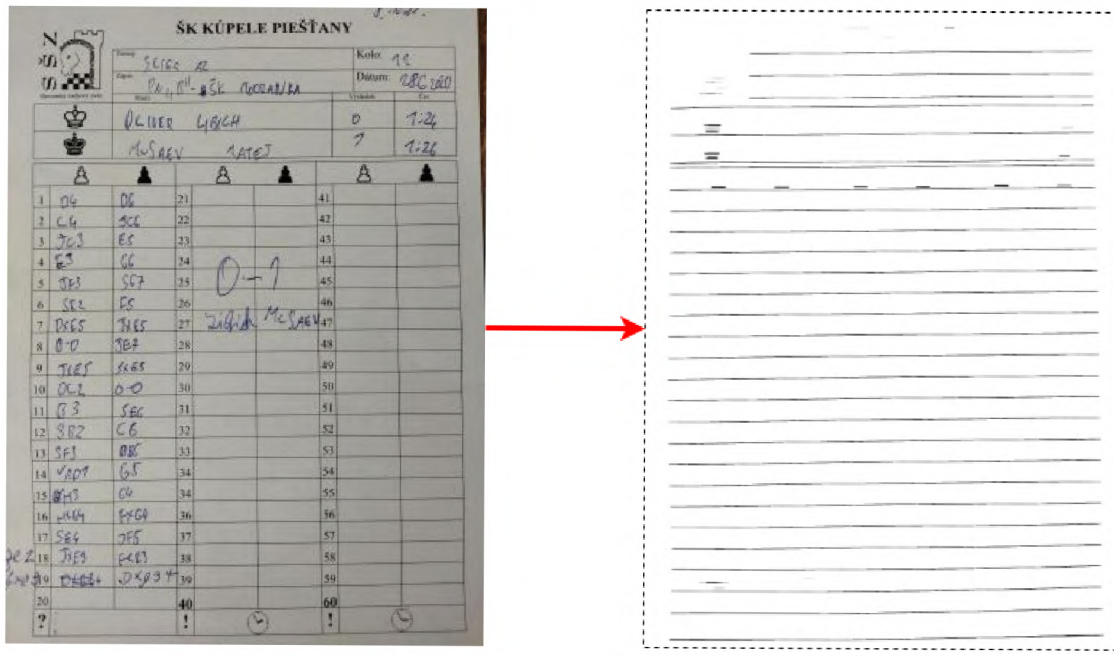
Na rozpoznanie tabuľky a buniek som vybral prístup na základe rozpoznania hraničných bodov tabuľky a jej následným vyňatím pomocou prespektívnej transformácie, pomocou horizontálnych a vertikálnych čiar. Na augmentáciu týchto čiar a ich následnú detekciu som vybral postup založený na morfologických operáciach a detekciu čiar pomocou Houghovej transformácie.

5.4 Vytvorenie kernelu

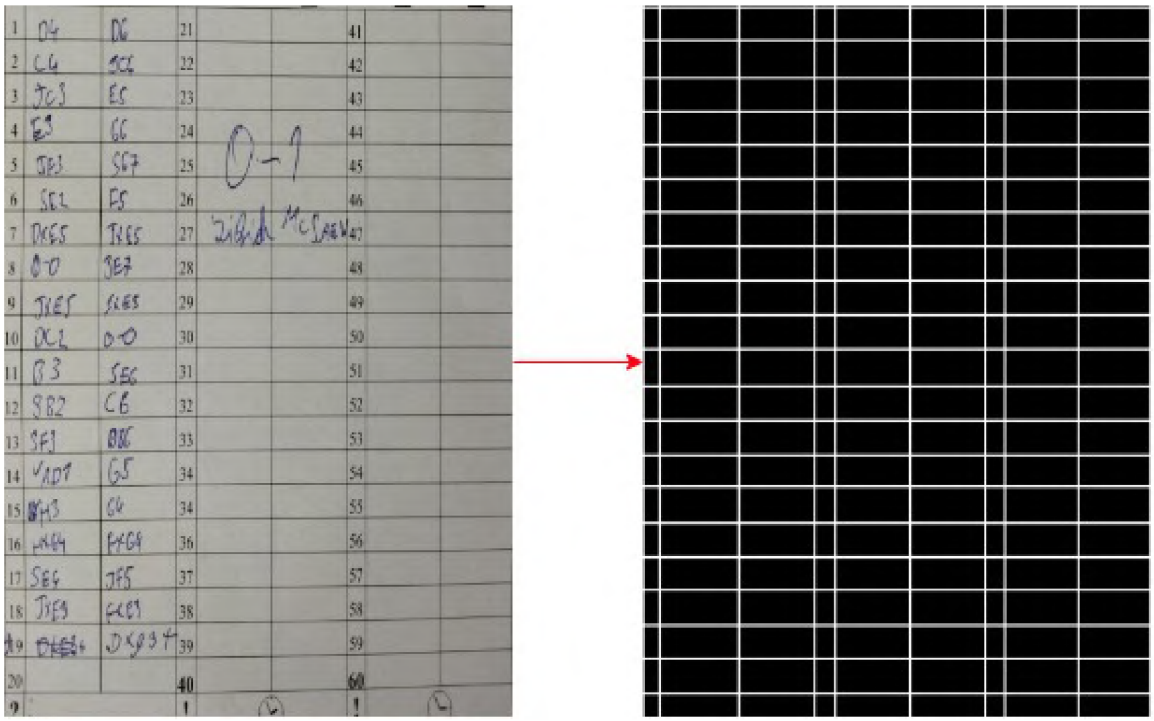
Na vytvorenie kernelu som vybral prístup podobný ako na zistenie hraničných bodov tabuľky. Najskôr sa augmentujú horizontálne a vertikálne čiary, ktoré sú identifikované Houghovou transformáciou. S týmito čiarami je potom postup nasledovný. Aj keď sú čiary na prvý pohľad spojené, jak je vidieť na obrázku 5.3, Houghova transformácia často detekuje čiary po malých celkoch. Pre uľahčenie práce je teda nutné tieto čiary zjednotiť. Keďže snímok je v tomto bode tabuľka ťahov so správnou rotáciou, dá sa usúdiť, že vertikálne

[1] `https://lichess.org`

aj horizontálne čiare sú približne rovné. Tento fakt je možné využiť tým, že všetky čiary, ktoré sa nachádzajú v tesnej blízkosti od seba napríklad ak priemerná hodnota súradníc x vertikálnej čiary je 50 a druhej 54, je možné tieto čiary spojiť do jednej. Týmto spôsobom sa potom všetky čiary v danej oblasti spoja do jednej čiary a vytvorí sa jedna čiara kernelu. Potom sa postupne prechádza cez všetky oblasti a z vytvorených čiar vznikne kernel, ktorý je možný vidieť na obrázku 5.4.



Obr. 5.3: Snímok po augmentácií horizontálnych čiar.



Obr. 5.4: Nájdený kernel tabuľky.

5.5 Spracovanie bunky

Hlavným cieľom spracovania bunky je odstránenie tabuľkových čiar ohraničujúcich oblasť na vpísanie ťahu. Výstupom by teda mal byť text pripravený na prevedenie cez OCR systém bez rušivých objektov. Žiadaný výstup je vidieť na obrázku 5.5. Mnohé partiáre nemajú ťahy zapísané presne do danej bunky a text často presahuje do okolitých buniek. Túto skutočnosť je nutné zohľadniť pri segmentácií snímku na bunky. Každá segmentovaná bunka z kernelu tabuľky je teda rozšírená o časť okolia.

Na odstránenie prebytočných čiar sa použil prístup, ktorý spočíva na podobnom princípe ako pri získavaní kernelu tabuľky zo sekcie 5.3. Na augmentáciu ohraničujúcich čiar boli využité morfologické operácie, nasledované rozpoznaním čiar Houghovou transformáciou. Tento prístup však takisto nie je bez problémov. Niektoré znaky v texte ako napríklad písmená **D** a **E** obsahujú v sebe vertikálne čiary, ktoré sú v niektorých prípadoch takisto augmentované. V niektorých prípadoch je riešením zväčšenie matice použitej pri morflogických operáciach. Avšak pri použitý príliš veľkej matice sú odstránené z rozpoznania aj ohraničujúce čiary. Aktuálnym riešením je filtrovanie nájdených čiar z bunky podľa ich pozícií. Ak sa nájdená čiara nachádza v strede snímku, daná čiara je ignorovaná. Po vyfiltrovaní čiar sú zostávajúce čiary odstránené. Výsledkom by mal byť čistý text na bielom pozadí s minimálnym počtom rušivých elementov. Výsledok je možné vidieť na obrázku 5.5.

Nie všetky bunky sú však bez rušivých elementov. V niektorých sa nachádzajú zvyšky ohraničujúcich čiar, ktoré neboli správne rozpoznané. Ukážka týchto buniek je na obrázku 5.6. Odstránenie týchto nežiadúcich zvyškov je potom spravené pomocou analyzovania spojených komponentov podľa farby. Komponenty sú považované za spojené, pokiaľ majú 2 priľahlé pixely rovnakú hodnotu. Po získaní týchto spojených oblastí sa vyfiltrujú podľa

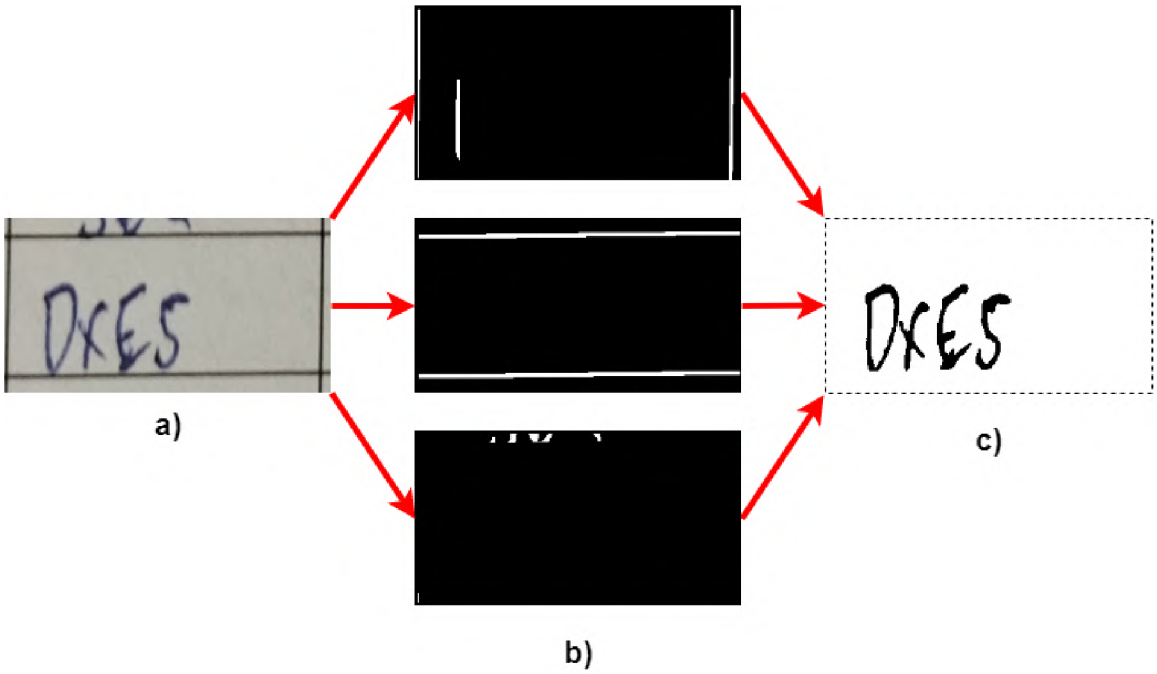
veľkosti, pričom oblasti s malým obsahom sú odstránené. Celý postup spracovania bunky je možné vidieť na obrázku 5.7.

Jeden z problémov, ktoré je potrebné vyriešiť je detekcia konca hry. Obvykle sa výsledok partie píše do náhodných voľných buniek v partiári, čo znamená že posledný zahratý ťah nemá žiadne špeciálne označenie, vďaka ktorému by sa dal ukončiť proces transkripcie. Na vyriešenie tohto problému sa využila kontrola čiernych pixelov v spracovaných bunkách. Pokiaľ má spracovávaná bunka malý počet čiernych pixelov, dá sa usúdiť, že sa v nej nenachádza žiadny text a predošlý spracovaný ťah bol posledným.

Tento prístup musí byť ešte upravený o počítanie spracovaných buniek v jednotlivých stĺpcoch. Návrh piešťanského partiára (pozri 2.1) obsahuje v jednom stĺpci 20 buniek pre ťahy a 1 bonusovú bunku pre zápis času. Táto bonusová bunka môže ale nemusí byť využitá. Aby sa predišlo problémom spojených s touto bonusovou bunkou, systém si uchováva súčasný počet spracovaných buniek v danom stĺpci. Pokiaľ sa počet dostane na číslo 20 bez zdetekovania prázdnej bunky, automaticky sa prejde na spracovanie nasledujúceho stĺpca.

Obr. 5.5: Ukážka textu z buniek bez rušivých elementov.

Obr. 5.6: Ukážka textu z buniek s rušivými elementami.



Obr. 5.7: **a)** Vstupná bunka s textom z obrázku. **b)** Detekcia a filtrovanie horizontálnych a vertikálnych čiar, detekcia rušivých elementov. **c)** Výstupná bunka bez ohraničujúcich čiar a rušivých elementov.

5.6 Rozpoznanie písma

Po spracovaní partiáru a rozdelení vstupného obrazu na jednotlivé bunky je možné pristúpiť k použitiu OCR systému. Tento bod je náročný najmä z dôvodu, že väčšina OCR systémov, ktoré pracujú offline, sú určené na strojom písané písmo. Aj keď niektoré tieto systémy sú schopné pracovať aj s rukou písaním písmom, úspešnosť ich predikcií nie je vysoká. Z tohto dôvodu bolo v prvotnom pláne otestovať predspracované snímky na dostupných systémoch, ktoré pracujú offline.

Prvým systémom, ktorý bol otestovaný je tesseract wrapper pre jazyk python – `pytesseract`. Prvotné výsledky nepriniesli veľkú mieru úspešnosti, keďže tento systém pracuje primárne so strojovo tlačeným písmom. Na zlepšenie výsledkov boli odskúšané rôzne techniky na zmenu vstupného textu v snímkoch, ako napríklad zmenšovanie/zväčšovanie hrúbky písma. Tieto pokusy však nepriniesli zlepšenie výsledkov.

Druhým testovaným systémom bol `easy-ocr`, avšak podobne ako `pytesseract`, aj tento systém primárne nepracuje s rukou písaným písmom. Výsledky dosahovali menšiu mieru úspešnosti ako `pytesseract` a transkripcie takisto trvala pomerne dlhšiu dobu.

Z týchto výsledkov bolo zrejmé, že je nutné použiť systém, ktorý je určený pre rukou písané písmo. V prvom rade bolo uvažované nad použitím online OCR systému ako napríklad pero-ocr[2], ktorý spadá pod projekt PERO[3] a je vyvíjaný na Vysokom učení technickom v Brne. Na tento nástroj boli nahraté testovacie snímky buniek, avšak aj keď bola úspešnosť predikcií vyššia ako spomínané offline nástroje, stále nebola dostatočne vysoká, aby čas strávený opravením zle vygenerovanými ťahmi nižší ako manuálny prepis partiárov.

[2] `https://pero-ocr.fit.vutbr.cz`
[3] `https://pero.fit.vutbr.cz`

Nakoniec sa od použitia online nástrojov upustilo najmä aj z dôvodu veľkého množstva požiadaviek, ktoré by z jednej hry vznikli. Pokiaľ hra skončila v 40. ťahu, obsahuje teda 40 ťahov bieleho a 40 čierneho, čo znamená že dokopy je potrebné zaslať najmenej 80 požiadaviek na vzdialený server, čo by zabralo dlhú dobu a nepotrebne zahlcovalo sieť. Jediným pokračovaním je teda natrénovanie vlastnej siete na rozpoznanie rukou písaného písma. Keďže neexistuje žiadna dostupná dátová sada ťahov slovenskej notácie, bolo nutné pred trénovaním neurónovej siete túto dátovú sadu vytvoriť. Výsledný systém je využiteľný najmä na tvorbu tejto dátovej sady, avšak jej použitie nebolo z časových dôvodov možné implementovať. O využití a rozšírení tejto dátovej sady, takisto aj postup nasledujúceho vývoja je popísaný v sekcií 7.4.

5.7 Postprocessing

Korekcia vygenerovaných ťahov je založená na kontrole znakov. Z analýzy valídnych ťahov som dospel k nasledujúcim faktom, na ktorých je potom založená kontrola :

Počet znakov	Postupnosť	Príklad ťahu
1	Neplatný ťah	e
2	Znak pola, číslica	b4
3	Znak figúry, znak pola, číslica	Je5
3	Značenie malej rošády	0-0
4	Znak figúry/pola, znak brania, znak pola, číslica	Jxe5
4	Znak figúry, znak pola alebo číslica, znak pola, číslica	Dae5
5	Znak figúry, znak pola/číslica, znak brania, znak pola, číslica	J1xb3
5	Značenie veľkej rošády	0-0-0
>6	Neplatný ťah	

Tabuľka 5.1: Tabuľka možných ťahov. Ak vygenerovaný ťah nespadá aspoň do jednej správnej kategórie, je vyhodnotený ako neplatný.

5.8 Uloženie transkripcie

Pre uloženie valídnej hry je nutné dodržať obecnú formu `pgn` súborov. Táto forma obsahuje niekoľko povinných polí, ktoré musia byť vyplnené (pozri 2.2). Vzhľadom na to, že tento systém nerieši hlavičku partiáru, z ktorého by bolo možne niektoré tieto informácie získať, majú všetky povinné polia uložených hier hodnotu `"?"`, poprípade `"*"` pre pole výsledku.

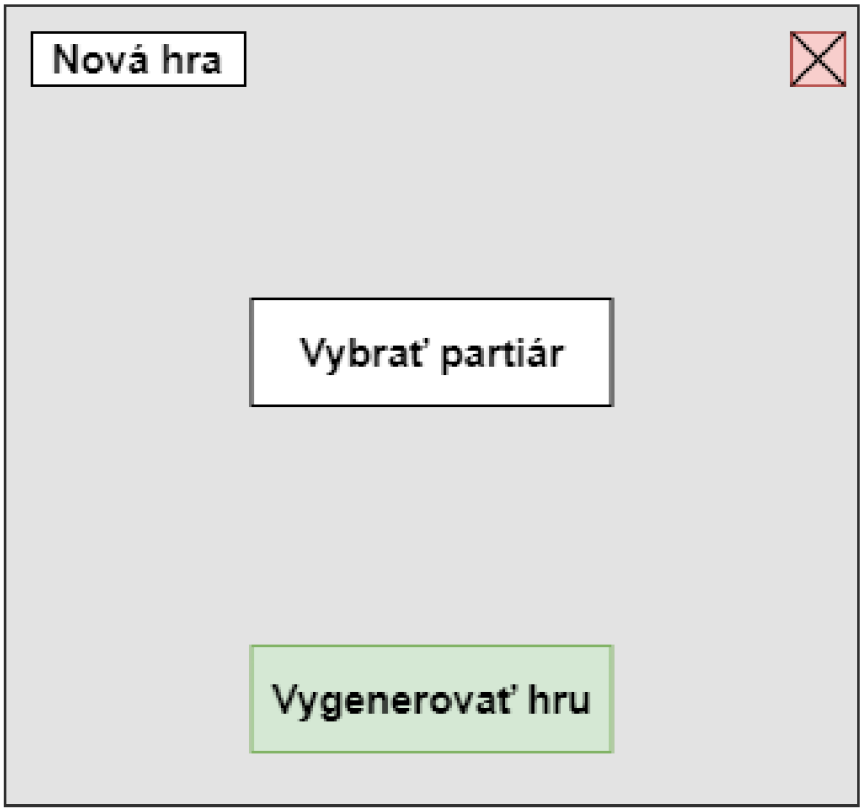
Keďže šachové programy určené k prezeraniu `pgn` súborov pracujú s anglickou šachovou notáciou, je nutné pred uložením `pgn` prepisu preložiť jednotlivé ťahy do anglickej notácie. Toto je jednoduchá operácia, ktorá spočíva v nahradením slovenských znakov pre figúry anglickými znakmi. Napríklad znak `S` (strelec) bude nahradený znakom `B` (bishop), znak `D` (dáma) znakom `Q` (queen) a tak ďalej.

5.9 Uživateľské rozhranie

Výsledok transkripcie každého partiáru musí byť valídna hra, ktorá neobsahuje neplatné ťahy. Samotný proces digitalizácie túto úroveň istoty nie je schopný poskytnúť, preto je

nutné vytvoriť aj uživateľské rozhranie, ktoré umožnuje užívateľovi opraviť zle vygenerované ťahy a skontrolovať samotnú hru pred jej uložením.

Uživateľské rozhranie pozostáva s dvoch okien. Hlavné okno slúži na výber partiáru a spustenie trankripcie. Druhé okno slúži na opravu vygenerovaných ťahov a uloženie tranksripcie. V tomto okne je dôležité dbať na farebné označenie zle vygenorovaných ťahov. Po každej úprave textu užívateľom sa aktualizuje farebné označenie textového pola. Pokiaľ text, ktorý sa zrovna nachádza v textovom poli je platný ťah, farebné podsvietenie sa zmení na zelenú, indikujúc správnosť ťahu. Táto kontrola je však čisto lexikálna, to či je daný ťah naozaj hrateľný v pozícií, ktorá je zrovna na šachovnici, systém nerieši. Tento bod musí kontrolovať sám užívateľ. Toto okno takisto poskytuje užívateľovi náhľad zadaného partiáru s označenými bunkami. Návrh okna pre výber partiáru je možné vidieť na obrázku 5.8.



Obr. 5.8: Návrh hlavného okna uživateľského rozhrania. Užívateľ bude schopný vybrať partiár a spustiť proces transkripcie.

Kapitola 6

Implementácia

V tejto kapitole popisujem mnou vytvorenú implementáciu systému. Digitalizácia šachových partiárov je málo preskúmanou oblasťou a existuje minimálne množstvo verejne dostupných vypracovaní tejto problematiky. Táto kapitola takisto popisuje zdôvodnenie vybraných postupov. Niektoré systémy pracujú s vlastným návrhom partiárov, poprípade s nahrávaním šablón a manuálnym označením priestoru pre vpisovanie ťahov. Tomuto postupu som sa však chcel vo vypracovaní vyhnúť, keďže mnoho hráčov nemá k dispozícií prázdny duplikát na nahranie šablóny. Vlastný návrh partiáru som takisto považoval za nesprávne smerovanie riešenia, najmä z dôvodu nevyužiteľnosti v praxi (zaškrtávací zoznam). Presadiť nový jednotný návrh partiáru by medzi klubmi bolo takisto náročné, pretože množstvo klubov má už vytlačené a pripravené partiáre na niekoľko rokov dopredu, to by znamenalo uvedenie digitalizátoru do praxe až s niekoľko ročným odstupom. Avšak je veľmi pravedepodobné, že nezavedenie týchto postupov má dopad na nízku úspešnosť výsledkov, ako ukazuje kapitola 7.

Výsledný systém je desktopová aplikácia s jednoduchým užívateľským rozhraním. Aplikácia umožnuje užívateľovi nahrať partiár, opraviť nesprávne rozpoznané ťahy a uložiť hru v pgn formáte.

6.1 Využité technológie

`Knižnice jazyka python`

- **OpenCV** - Python knižnica na spracovanie problémov spojených s počítačovým videním. Poskytuje rozhranie na prácu a úpravu rôznych typov dokumentov, analýzu štruktúry snímkov, videí a podobne.
- **imutils** - Python knižnica na jednoduché spracovanie snímkov, ktorá poskytuje potrebné funkcie ako napríklad rotáciu, zväčšenie/zmenšenie snímku, detekciu hrán a mnohé iné.
- **Chess pgn** – Knižnica poskytujúca potrebné funkcie na čítanie a parsovanie šachových hier v štandardnom formáte pgn. Na prechádzanie šachovej partie je konvertovaný pgn formát na formát FEN.
- **Python-chess** – Podobne ako Chess pgn, aj táto knižnica slúži na prácu s pgn súbormi. Avšak výhoda oproti Chess pgn je možnosť prechádzania po jednotlivých ťa-

hoch bez toho, aby boli konvertované na FEN notáciu. Vysvetlenie FEN notácie je možné nájsť tu [6].

- **Tkinter** – Python knižnica na tvorbu grafických uživateľských rozhraní. Je založená na Tcl/Tk nástrojoch a ponúka rôzne grafické prvky a nástroje pre tvorbu uživateľských rozhraní. Tkinter je multiplatformová knižnica, pričom programy v ňom vytvorené sú schopné fungovať na rôznych operačných systémoch [7].
- **Pytesseract** – Nástroj pre optické rozpoznanie znakov pre python. Pytesseract je wrapper na Tesseract-OCR[1] od spoločnosti Google. Dokáže pracovať s rôznymi typmi snímkov ako napríklad `jpeg`, `png`, `tiff` a iné.

6.2 Spracovanie partiárov

Celý proces digiatlizácie partiáru je spravený vo funkcii `transcribe_scoresheet`, ktorá má ako parameter cestu ku snímku a vracia dvojicu polí vygenerovaných ťahov bieleho a čierneho hráča. Po načítaní a prevedení snímku do binárnej podoby je prvým krokom opraviť zlú rotáciu snímku. Vytvorí sa štruktúrovacia matica na augmentáciu horizontálnych čiar pomocou funkcie knižnice opencv (ďalej len cv2) `cv2.getStructuringElement`. Následne sú na snímok aplikované morfologické operácie erózie a dilatácie. Tieto operácie takisto poskytuje knižnica opencv a to vo funkciách `cv2.erode` a `cv2.dilate`. Z takto upraveného snímku sa následne rozpoznajú čiary pomocou `cv2.HoughLinesP`. Na zvolené parametre sa prišlo experimentovaním s rôznymi hodnotami. Pomocou numpy polí sa vypočíta priemerný sklon vygenerovaných čiar. Následne je tento sklon použitý na výpočet uhlu rotácie. Pomocou `imutils.rotate`, ktorý má ako parameter zistený uhol sa orotuje obraz. Po tomto kroku sa pracuje s vyrovnaným partiárom.

Ďalším krokom je nájdenie hlavnej tabuľky na zapisovanie ťahov. V tomto bode sa dá využiť využiť nájdenie jednej zo šestice šachových figúr v partiári, keďže tieto grafické prvky sú priamo nad tabuľkou s ťahmi. Na tento úkon sa použila funkcia `cv2.findContours`. Táto funkcia sa používa na lokalizáciu spojitých oblastí v obraze [8]. Výsledkom tejto metódy je zoznam kontúr, ktoré sú reprezentované bodmi, tvoriacimi jej hranice. Takisto ide získať ďalšie dôležité informácie ako obsah kontúry, čo je dôležité pri ďalšej práci. Následne sa nájdené kontúry filtrujú podľa obsahu, pozícií na snímku a počtom bodov v jej hraniciach. Po nájdení kontúry, ktoré splňujú požadované vlastnosti, sa snímok oreže podľa jej spodnej hrany. Po tomto kroku sa pracuje s partiárom, ktorý neobsahuje dodatočné informácie v hlavičke.

Nasledujúcim krokom je získanie hraničných bodov tabuľky na použitie perspektívnej transformácie. Keďže všetky 4 hraničné body by mali byť súčasťou najvonkajších čiar tabuľky, znovu sa využije prístup založený na augmentácií vertikálnych a horizontálnych čiar a následným aplikovaním Houghovej transformácie. Z výstupu Houghovej transformácie sa potom nájdu 4 body, ktoré majú najnižšiu euklidovskú vzdialenosť od rohov snímku. To znamená, že sa nájdu najbližšie body k bodom `(0,0)`, `(0, výška_snímku)`, `(šírka_snímku, 0)` a `(šírka_snímku, výška_snímku)`. Nájdené body sú použité na perspektívnu transformáciu. Na to slúži dvojica funkcií `cv2.getPerspectiveTransform`, ktorá slúži na vytvorenie transformačnej matice a `cv2.warpPerspective`, ktorá transformuje snímok za použitie získanej matice.

[1] `https://github.com/tesseract-ocr/tesseract`

Po získaní hlavnej tabuľky nasleduje vytvorenie kernelu na extrakciu jednotlivých buniek. Po aplikovaní morfologických operácií a rozpoznaní čiar sa úseky čiar v rovnakej oblasti spoja do jednej a vytvoria jednu čiaru kernelu. Získanie čiar kernelu sa nachádza vo funkcií `get_static_lines`. Princíp tejto funkcie spočíva v zobrazovaní a kontrole jednotlivých rozpoznaných úsekov čiar. Pokiaľ sa súčasná čiara nachádza v tesnej blízkosti už zobrazenej čiary, táto čiara sa ignoruje. Pokiaľ ešte zobrazená nie je, vytvorí sa nová čiara, ktorá je vedená buď od výšky 0 až do výšky snímku so súradnicou x súčasnej čiary alebo od 0 až do šírky snímku so súradnicou y súčasnej čiary. Proces vytvorenia jednotlivých čiar je delení na spracovanie horizontálnych a vertikálnych čiar. Z vytvorených čiar sú nakoniec spravené ich intersekcie vo funkcií `find_cross_sections`. Každé 4 body intersekcie popisujú jednu bunku tabuľky, pričom je nutné rozlíšiť, či sa jedná o bunku s textom ťahu alebo bunku s číslom ťahu. Nájdené bunky sú potom spracovávané po stĺpcoch, začínajúc zľava. Pre kontrolu, či je spracovávaný stĺpec text ťahu, alebo číslo sa využíva obsah prvej bunky stĺpcu. Pokiaľ platí :

$$S_{bunka} >= S_a - \frac{S_a * 2,5}{5}; S_a = \frac{S_{snimok}}{120} \tag{6.1}$$

kde S_{bunka} je obsah skúmanej bunky v partiári a S_{snimok} je obsah celého vstupného obrázka, tak nájdená bunka je označená ako bunka s textom ťahu a spracuje sa stĺpec tabuľky. Inak sa súčasná bunka označí ako bunka s číslom ťahu a spracovanie stĺpcu sa preskočí. Táto rovnica vychádza zo skutočnosti, že počet všetkých buniek s ťahmi by mal byť 120, pričom bunky s ťahom sú približne 4 krát väčšie ako bunka s číslom ťahu.

Spracovanie buniek jednotlivých stĺpcov začína odstránením ohraničujúcich čiar. Morfologickými operáciami sa augmentujú ohraničujúce čiary a potom sa rozpoznajú Houghovou transformáciou. Po prevedení bunky na čiernobiely obraz pomocou `cv2.threshold`, ktorý má určený typ prahovania ako Otsu prahovanie (`cv2.THRESH_OTSU`) sa odstránia nájdené čiary. Čiary sú odstránené pomocou funkcie `cv2.line`, ktorá zmení farbu všetkých nájdených čiar na bielu.

Ďalším krokom je odstránenie zvyškových rušivých elementov. Na rozpoznanie spojených oblastí sa využila funkcia `cv2.connectedComponentsWithStats`, ktorá okrem ohraničujúcich bodov oblastí vracia aj jej obsah, ktorý sa využije pri filtrovaní rušivých elementov od textu. Porovnaním s obsahu sa vyfiltrujú malé oblasti, ktoré sú označené ako rušivé a následne sú z obrazu odstránené. Bunka je v tomto bode pripravená na rozpoznanie písma.

Detekcia plnosti buniek prebieha vo funkcií `cell_not_empty`, ktorá berie ako argument binárny snímok spracovanej bunky. Vypočíta sa počet všetkých pixelov v snímku pomocou výšky a šírky snímku a spočíta sa počet všetkých bielych pixelov pomocou funkcie `cv2.countNonZero`. Počet čiernych pixelov v snímku je potom vyrátaný odčítaním bielych pixelov od celkového pixelov v celej bunke. Funkcia vratia hodnotu `True` pokiaľ počet čiernych pixelov je väčší ako 0,5 % všetkých pixelov bunky. Funkcia vracia hodnotu `False`, ak toto porovnanie neplatí.

Ako systém pre rozpoznanie písma bol využitý voľne dostupný pytesseract. Jeho funkcia `pytesseract.image_to_data` vracia rozpoznaný text a takisto aj koeficient istoty. Vďaka možnosti konfigurácie tohto nástroja bolo možné nastaviť povolené znaky, takisto aj zrušiť používanie jazykového modelu pre zlepšenie predikcií. Keďže šachová notácia obsahuje slová, ktoré sa nenachádzajú v rôznych dostupných jazykových modeloch, bola aj táto možnosť použitá. Po skončení rozpoznania textu sa vygenerovaná postupnosť znakov uloží do pola a pokračuje sa ďalšou bunkou.

Postprocessing a korekcia vygenerovaných ťahov je implementovaná v súbore `moves_validator.py`. Niektoré funkcie ako napríklad `convert_to_svk` alebo `convert_to_eng` slúžia na preklad ťahov medzi anglickou a slovenskou notáciou. Funkcia `normalize_move` slúži na normalizáciu ťahov prevodom zle vygenerovaných veľkých a malých znakov. Počas transkripcie sa môžu objaviť neplatné znaky, ktoré sa však dajú jednoducho spracovať. Príkladom takéhoto znaku je napríklad veľké `G`, ktoré sa nemôže nachádzať v `pgn` formáte. Avšak prevodom znaku `G` na malé `g` sa dá dostať k valídnemu označeniu políčka na šachovnici. V tomto súbore je takisto implementovaný validátor ťahov vo funkcií `move_is_valid`. Podľa dĺžky vygenerovaného textu a prechádzaním jednotlivých znakov určuje, či vstupný text je valídny šachový ťah alebo nie. Validácia ťahov je spravená podľa tabuľky 5.1.

6.3 Užívateľské rozhranie

Užívateľské rozhranie bolo implementované pomocou knižnice `tKinter`. Po spustení aplikácie sa otvorí hlavné okno, ktoré je definované v triede `MainWindow`. Táto trieda definuje niekoľko metód na prácu s pgn súborom. Metóda `choose_pgn` sa spúšťa po kliknutí na tlačidlo pre výber partiáru. Užívateľovi sa otvorí dialógové okno, z ktorého je schopný vybrať požadovaný súbor. Po vybratí partiára môže užívateľ začať proces digitalizácie kliknutím príslušného tlačítka.

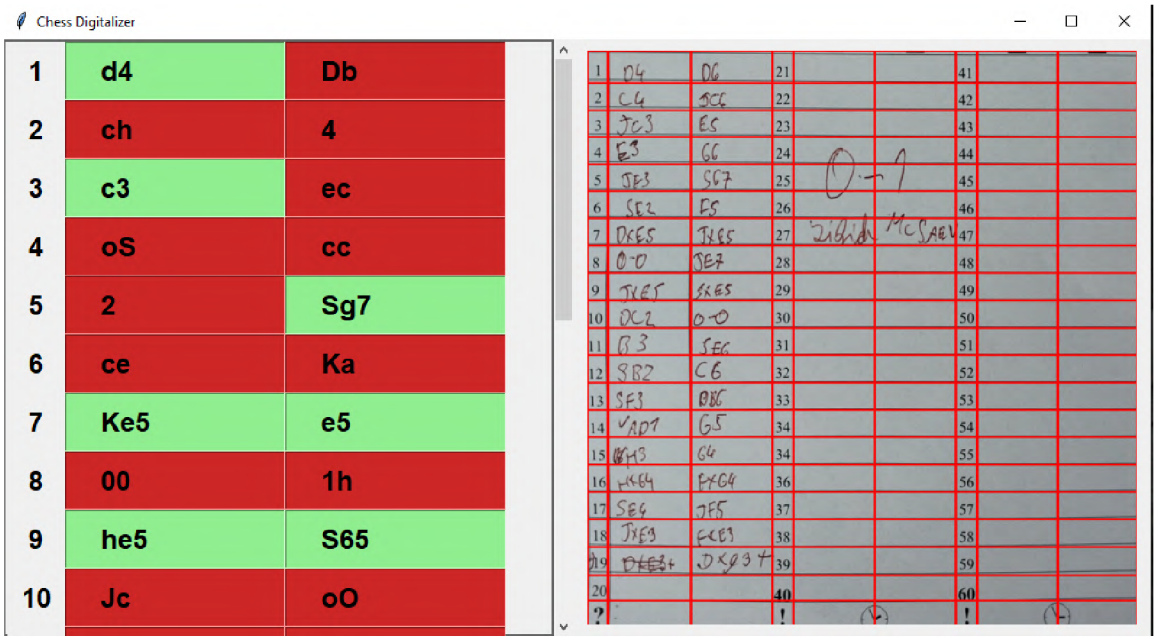
Spracovanie partiáru rieši metóda `open_pgn_window`. Po skončení digitalizácie je otvorené druhé okno, v ktorom sú ukázané vygenerované ťahy. Každé textové pole je triedy `CustomText`, ktorá rozširuje pôvodnú triedu `tk.Text`. Táto trieda bola prevzatá z [18]. Toto rozšírenie spočíva vo vygenerovaní udalosti `«TextModified»` pokiaľ v danom textovom poli prišlo k zmene či už odobratím, pridaním alebo zmenením textu. Toto rozšírenie pomáha pri prideľovaní farebného odlíšenia jednotlivých ťahov. Po každej udalosti `«TextModified»` sa zavolá validačná funkcia `move_is_valid`, ktorá skontroluje súčasný text v textovom poli, ktoré vygenerovalo túto udalosť, a vyhodnotí či text je platný alebo neplatný šachový ťah. Pokiaľ je ťah, ktorý sa zrovna nachádza v textovom poli neplatný, je označený červenej farbou, čo nabáda užívateľa k opraveniu tohto ťahu. Valídny ťah je označený farbou zelenou.

Po kontrole a oprave ťahov môže užívateľ uložiť hru v pgn formáte. Po kliknutí na príslušné tlačítko sa otvorí dialógové okno a užívateľ môže vybrať adresár na uloženie a názov vytvorenej hry. Po uložení môže užívateľ pokračovať v digitalizácií ďalšieho partiáru. Pre jednoduchšie opravovanie zle vygenerovaných ťahov je užívateľovi v tomto okne aj ukázaný nahratý partiár so zvýraznenými segmentovanými bunkami. Užívateľ tak nemusí hľadať ťahy v partiáry v papierovej podobe a môže sa lepšie sústrediť na opravovanie zle vygenerovaných ťahov. Výsledné hlavné okno je možné vidieť na obrázku 6.2 a okno na skontrolovanie transkripcie na obrázku 6.1.

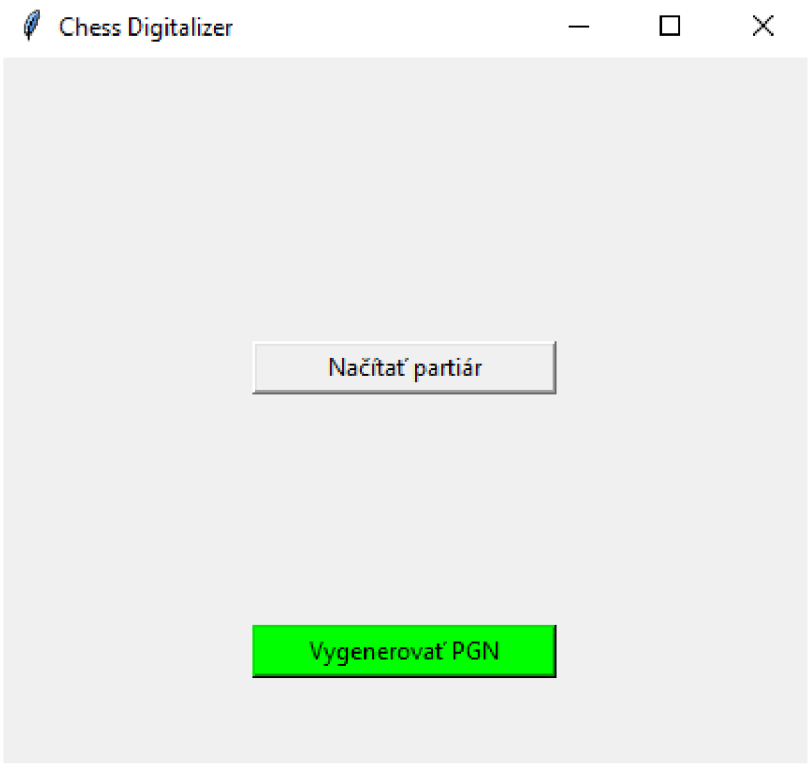
6.4 Testovanie

Na testovanie systému a vyhodnotenie úspešnosti rozpoznávania textu bol vytvorený testovací skript s názvom `test.py`. Upravením premennej `folder` sa dá vybrať testovaná zložka. Testovací skript na svoju prácu využíva pripravené `pgn` súbory, z ktorých pomocou funkcií knižníc `chess.pgn` a `pgn_parser` vyťahuje jednotlivé ťahy, ktoré slúžia na vyhodnotenie úspešnosti rozpoznania znakov. Skript jednotlivo prechádza všetky partiáre v danej zložke a pomocou funkcie `transcribe_scoresheet` dostane vygenerovaný list ťahov bieleho

a čierneho hráča, ktoré sú potom porovnávané s ťahmi v predpripravenom `pgn` súbore celej hry. Skript po prejdení každého partiáru vypíše informácie ako napríklad priemernú úspešnosť rozpoznania ťahov. Po prejdení všetkých partiárov skript vypíše priemernú úspešnosť rozpoznania ťahov všetkých partiárov v danej zložke.



Obr. 6.1: Ukážka okna na opravu chýb vygenerovaných ťahov. Užívateľ môže opraviť zle vygenerované ťahy a skontrolovať celú transkripciu. Užívateľ takisto vidí náhľad partiáru a výsledok segmentácie buniek.



Obr. 6.2: Hlavné okno uživateľského rozhrania. Užívateľ môže zvoliť partiár a začať proces digitalizácie.

Kapitola 7

Vyhodnotenie výsledkov

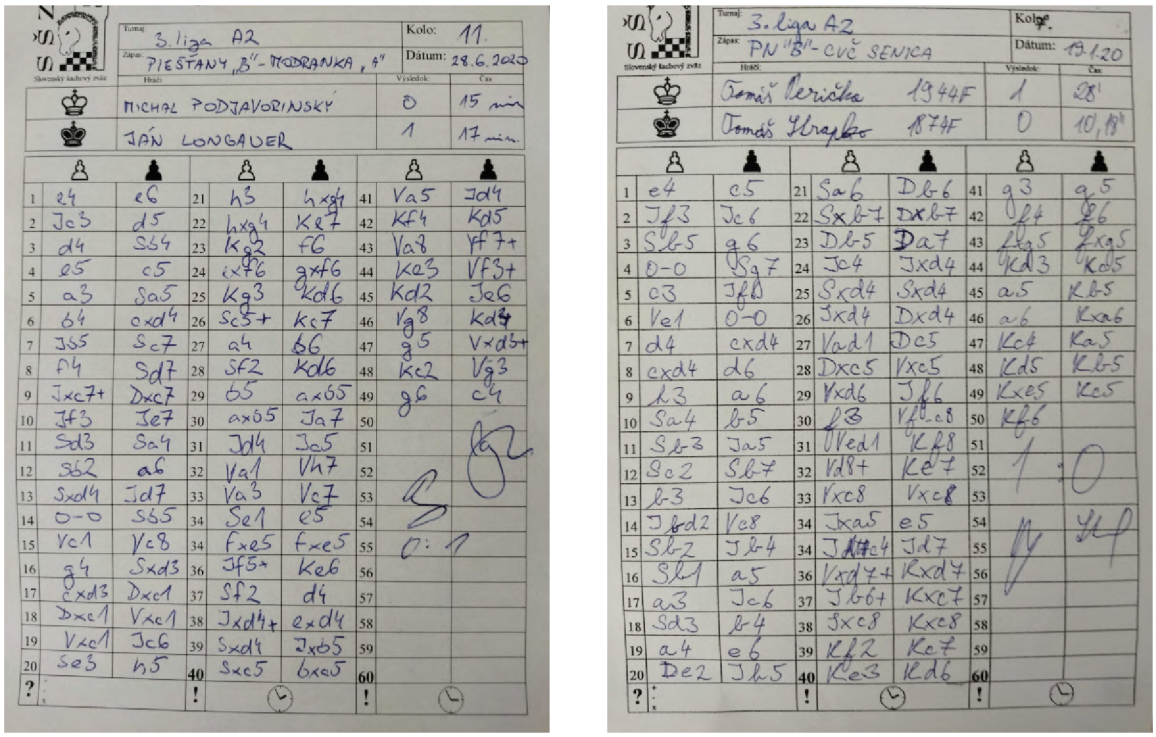
Táto kapitola oboznamuje čitateľa s využitou dátovou sadou na testovanie, procesom testovania jednotlivých častí vytvoreného systému, výsledkami testovania a na konci kapitoly sú popísané možnosti budúceho vývoja.

7.1 Testovacia dátová sada

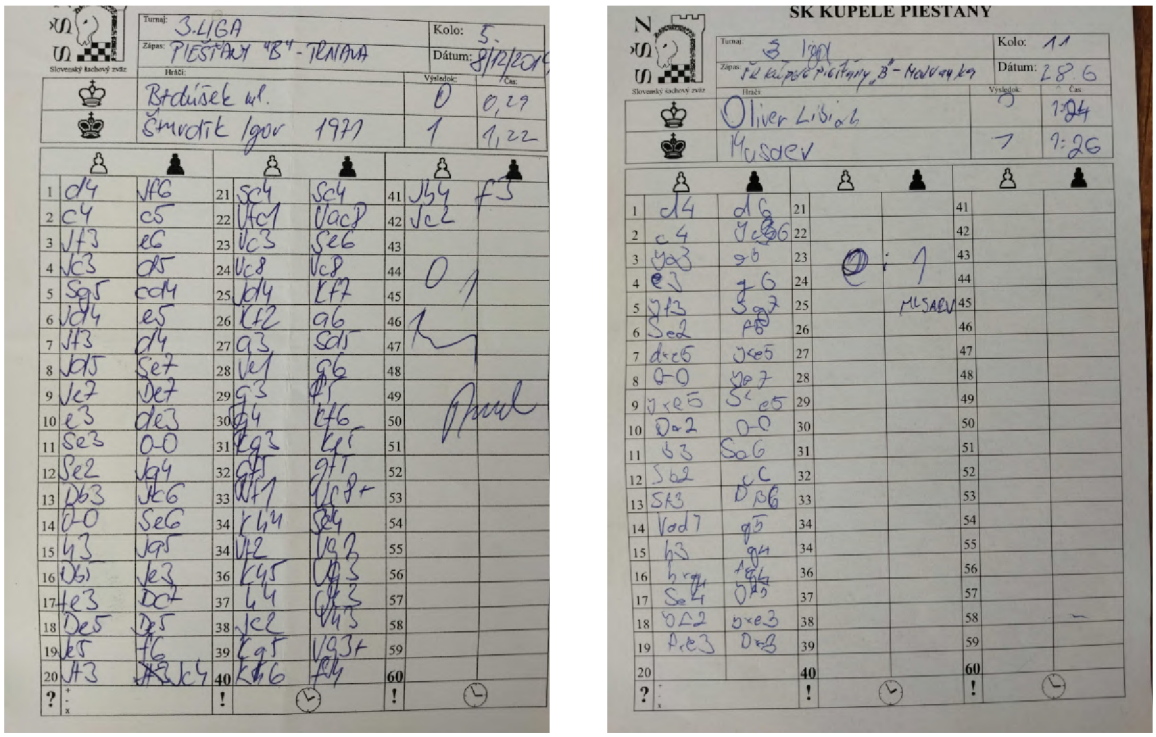
Dátová sada použitá na testovanie funkčnosti vytvoreného programu pozostáva z dvojice partiárov jednej hry a jej výsledného PGN. Všetky partiáre sú rovnakého formátu, ktorý využíva šachový klub Piešťan. Plán pre nasledujúci vývoj je rozšírenie počtu hier z piešťanského klubu, takisto aj získanie partiárov z iných klubov. Pre lepšiu evaluáciu očakávaných výsledkov, budú partiáre rozdelené do nasledujúcich kategórií.

- **A** – Toto sú partiáre najvyššej kvality. Písmo je dobre čitateľné, obsahujú minimálne množstvo chýb, ťahy majú minimálny presah do okolitých buniek. Ukážka je na 7.1.
- **B** – Toto sú partiáre zníženej kvality. Písmo je ťažšie čitateľné, obsahuje malé množstvo chýb, niektoré ťahy majú presah do okolitých buniek. Ukážka je na 7.2.
- **C** – Toto sú partiáre najhoršej kvality. Písmo je veľmi ťažko čitateľné, môže obsahovať množstvo chýb. Ukážka je na 7.3.
- **D** – Toto sú partiáre, ktoré obsahujú v ťahoch dodatočné informácie, ako napríklad čas na hodinách po odohratí ťahu. Ukážka je na 7.4.

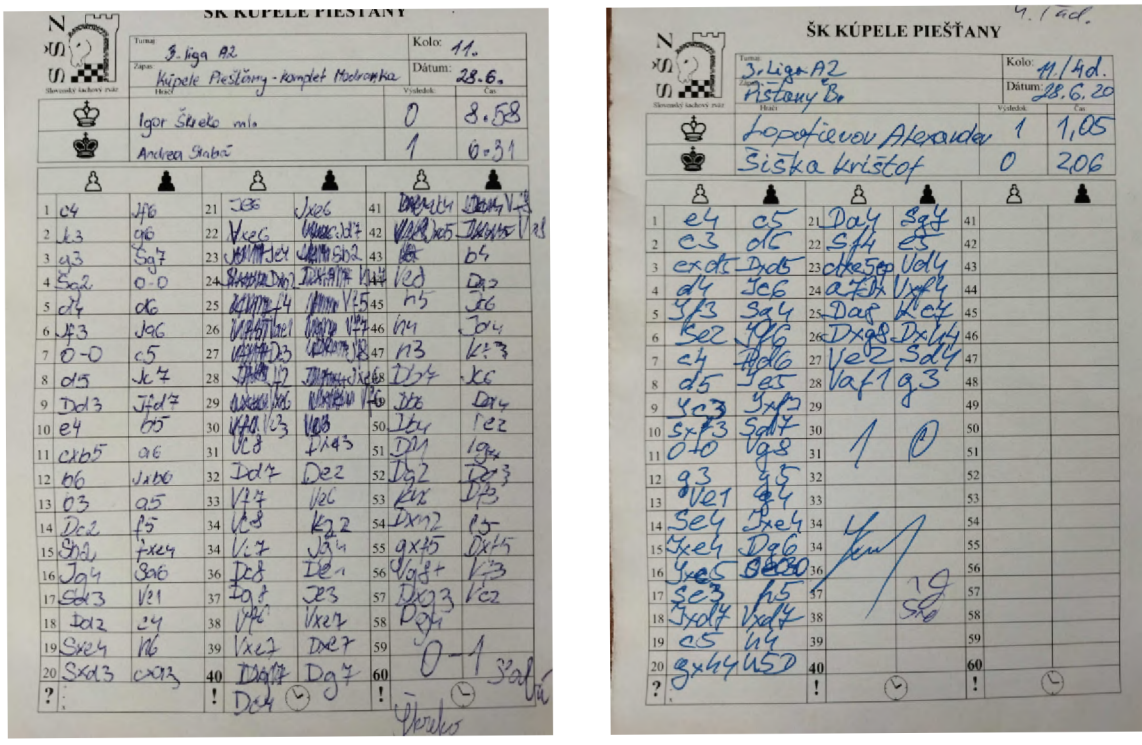
Chyby v partiároch sú napríklad zle zapísané ťahy, vynechanie ťahu, preškrtnuté ťahy, ťahy zapísané mimo danej bunky a podobné. V súčasnej dobe obsahuje testovacia dátová sada 21 šachových hier, pričom jedna z hier obsahuje len jeden partiár. Celkový počet partiárov je teda 41 a počet ich prepisov v `pgn` formáte 21.



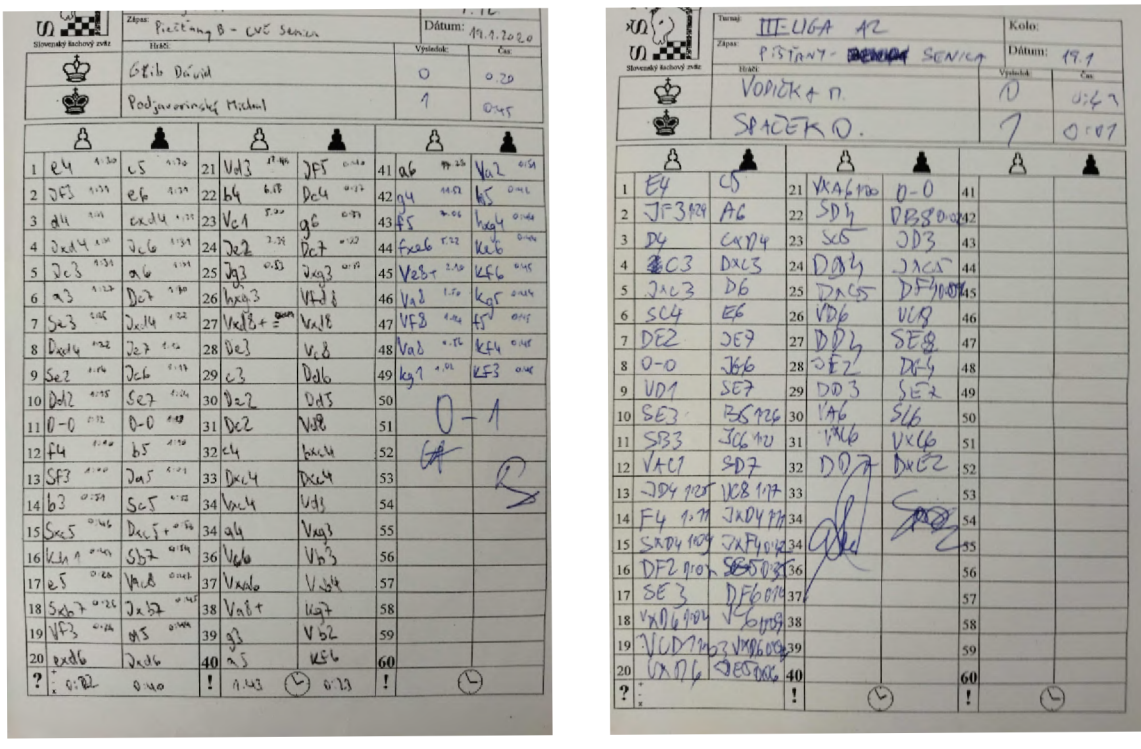
Obr. 7.1: Ukážka partiárov z kategórie A. Písmo je dobre čitateľné a ťahy majú malý presah do okolitých buniek.



Obr. 7.2: Ukážka partiárov z kategórie B. Písmo je ťažšie čitateľné a niektoré ťahy presahujú do okolitých buniek.



Obr. 7.3: Ukážka partiárov z kategórie C. Písmo je ťažko čitateľné a v partiároch sa môže nachádzať množstvo chýb.



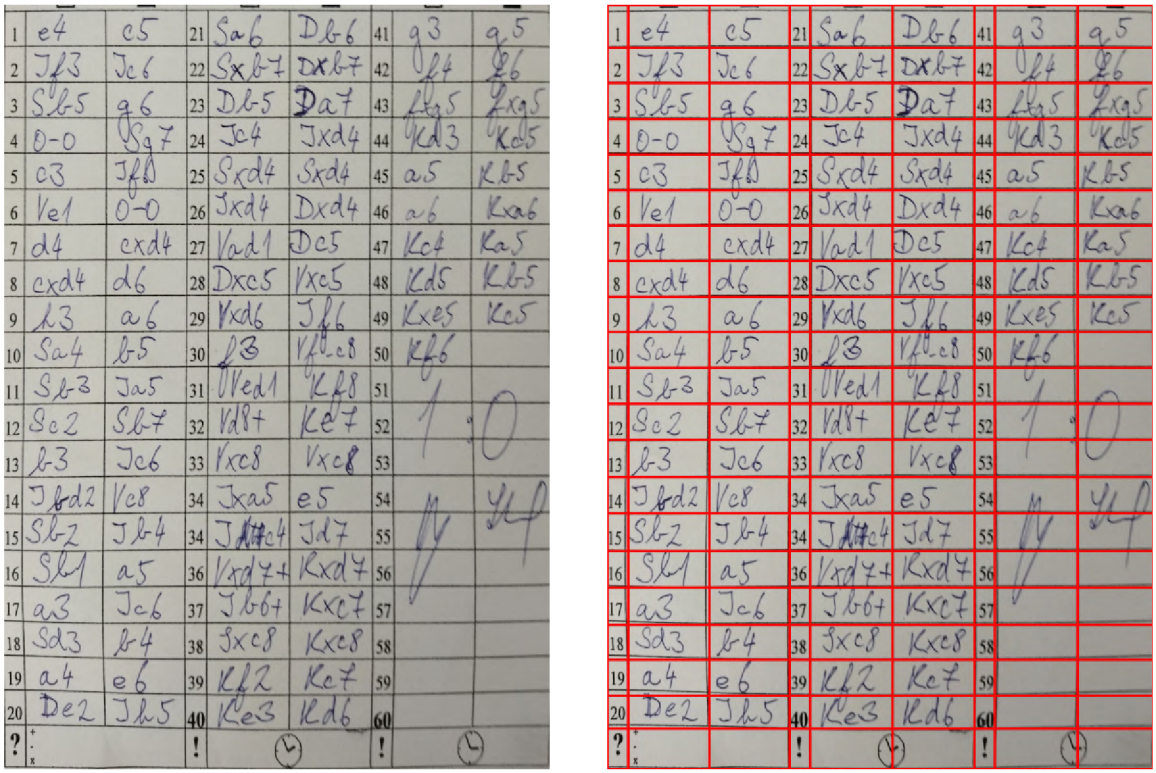
Obr. 7.4: Ukážka partiárov z kategórie D. Súčasťou niektorých ťahov sú aj nadbytočné informácie ako napríklad čas spotrebovaný pri zahranom ťahu.

7.2 Segmentácia buniek

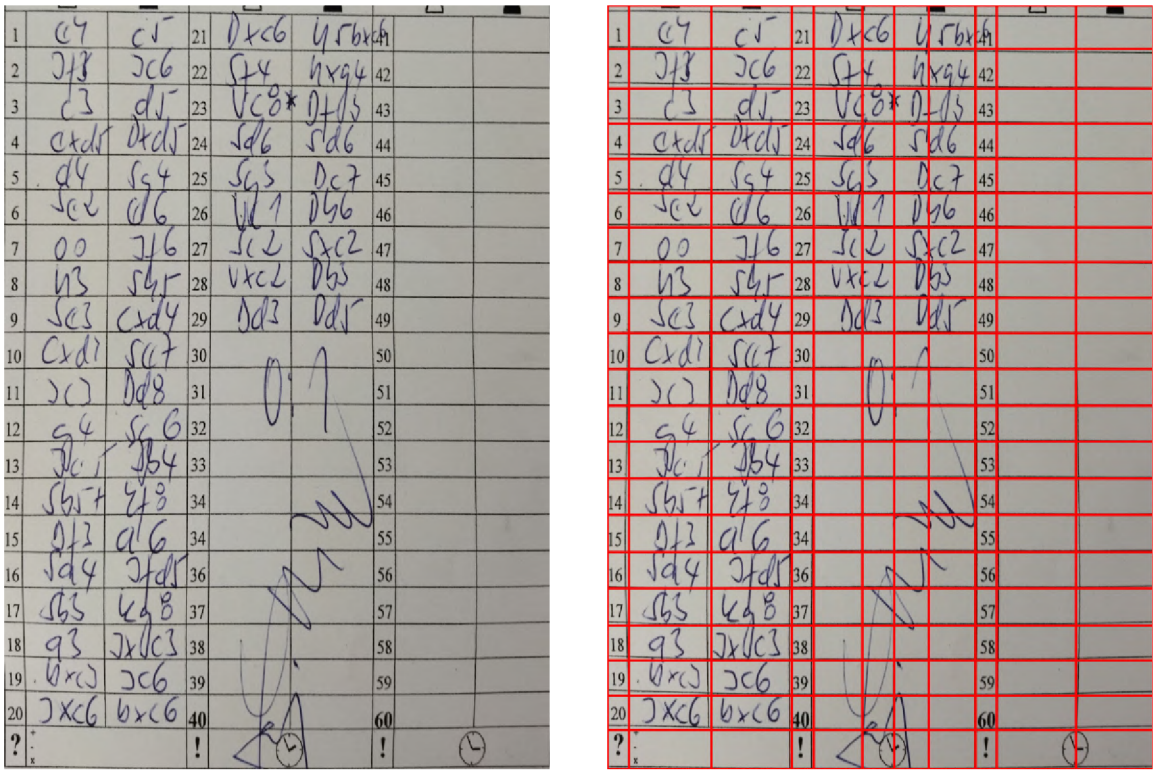
Vyhodnotenie správnosti segmentácie buniek prebiehalo manuálnou kontrolou. Pre tento účel bola upravená funkcia na generovanie tranksripcie, pričom končila zobrazením snímku so zvýrazneným kernelom tabuľky. Nájdený kernel tabuľky bol v snímkoch partiárov zvýraznený červenou farbou s dostatočnou hrúbkou, aby bol na fotke dobre viditeľný. S takto upraveným skriptom sa skontrolovali partiáre pričom sa prihliadalo na počet správne vyznačených polí s ťahmi v partiároch. Pokiaľ by niektoré bunky označené neboli vôbec, alebo dvojica buniek spojená do jednej, počítalo by sa to ako chybu. Po prejdení 41 partiárov je pre rôzne kategórie úspešnosť segmentácie popísaná tabuľkou 7.1.

Kategória	Úspešnosť
A	100 %
B	97,2 %
C	80,51 %
D	100 %

Tabuľka 7.1: Porovnanie výsledkov segmentácie podľa katégorie dátovej sady.



Obr. 7.5: Výsledok segmentácie so 100 % úspešnosťou. Každá bunka je správne segmentovaná.



Obr. 7.6: Neúspešný výsledok segmentácie. Podpis hráča na partiáry vyhodnocuje systém ako jednu z vertikálnych čiar a bunky v strednom stĺpci sú nesprávne rozdelené na 4 časti.

7.3 Rozpoznanie textu

Hlavným výstupom tejto práce je systém, ktorý slúži na digitalizovanie šachových partiárov a zároveň vytvára dátovú sadu na trénovanie neurónovej siete pre rozpoznanie ručne písaných ťahov šachu. Pôvodný plán vytvoriť systém, ktorý bude schopný tento prepis robiť automaticky s minimálnymi opravami človeka sa nestihol zrealizovať. V súčasnej forme musí človek opraviť viac ako 80 % vygenerovaných ťahov. Problém nastáva v samotnom systéme pre rozpoznanie textu. Aj keď je ako vstup použitý partiár, ktorý dosahuje 100 % úspešnosť segmentácie buniek, OCR systém `pytesseract` nedokáže vo väčšine prípadoch správne rozpoznať obsiahnutý text. Pomocou predspracovania jednotlivých buniek sa podarí pripraviť na vstup OCR systému binárny snímok s textom, ktorý neobsahuje rušivé elementy, ale výsledky transkripcie nie sú správne. Ukážky vstupov rozpoznania textu sú na obrázku 7.7.

Testovanie úspešnosti rozpoznania ťahov prebiehalo s voľne dostupným nástrojom pytesseract. Testovanie prebehlo za použitia pripraveného datasetu, pričom boli vyhodnocované jednotlivé kategórie zvlášť. Porovnanie výsledkov je možné nájsť v tabuľke 7.2. Na určenie miery úspešnosti prepisu jednotlivých ťahov bola použitá technika Levenshteinovej vzdialenosti. Táto technika vypočíta počet operácií, ktoré je potrebné vykonať, aby sa z textu `A` stal text `B`. Podobnosť dvoch textov sa teda dá vypočítať nasledovne :

```
def levenshtein_percentage(a, b):
    dist = levenshtein_distance(a, b)
    len_a = len(a)
    len_b = len(b)

    return 1 - (dist / max(len_a, len_b))
```

Ak by vygenerovaný text bol `JfG` a cieľový text `Jxf5`, miera úspešnosti by bola 50 %, keďže by sa museli vykonať 2 operácie – zmena znaku `G` na číslicu `5` a pridanie znaku `x`.

Kategória	Úspešnosť
A	16,23 %
B	12,21 %
C	11,51 %
D	9,11 %

Tabuľka 7.2: Porovnanie úspešnosti rozpoznania znakov dátových kategórií.

Podľa očakávaní dosiahli najlepšiu mieru výsledkov snímky v kategórií `A` a to 16,23 %. Aj keď systém nemá vysoké výsledky ani pre partiáre dobrej kvality, toto porovnanie ukazuje, že čitateľnosť písma je jeden z dôležitých faktorov na správne rozpoznanie písma. Takisto je nutné zdôrazniť, že pri niektorých prípadoch systém nemá šancu správne vyhodnotiť daný ťah. Takéto prípady sú napríklad zaškrtané bunky, ako je vidieť na jednom z partiárov z kategórie D (pozri 7.4) alebo je písmo nerozpoznateľné ani človekom. Snímky týchto ťahov sú na obrázku 7.8

Obr. 7.7: Ukážka vstupov binárnych snímkov do OCR systému. **A)** Vstupom je text `c6` a výstup je totožný (`c6`). **B)** Vstupom je text `Dxc5` a výstup je dvojica znakov `DS`. **C)** Vstupom je text `d4` a výstup je číslica `4`. **D)** Vstupom je text `Se3` a výstupom je dvojica číslic `54`.

Obr. 7.8: Nečitateľné ťahy v partiároch. **A)** `Je7`. **B)** `e6`. **C)** `Dxe3`. **D)** `Jc3`.

7.4 Možnosti budúceho vývoja

Ako bolo spomenuté v predchádzajúcich kapitolách, výsledný systém nemá vysokú úspešnosť rozpoznania textu v jednotlivých bunkách partiára. Nasledujúci postup by mal byť zameraný najmä na tento problém. Z minulých sekcií vyplýva, že použitie voľne dostupných nástrojov na rozpoznanie textu je pre takýto špecifický problém nedostatočné riešenie, preto je nutné vytvoriť vlastný nástroj, ktorý bude tohto schopný. Na vytvorenie tohto nástroja môže pomôcť dátová sada šachových ťahov[1], ktorá bola vyprodukovaná za použitia vytvoreného systému v tejto bakalárskej práci. Táto dátová sada obsahuje 1915 anotovaných snímkov jednotlivých buniek šachových partiárov. Po vytvorení systému na rozpoznanie písma, je tu ešte mnoho prístupov, ktoré vedia zlepšiť úspešnosť predikcií. Niektoré z nich sú spísané v nasledujúcich bodoch:

- **Predikcie na základe knihy otvorení**. Jedna z techník môže využiť zavedenie knihy otvorení na zlepšenie predikcií prvých ťahov. Šachová partia sa neoficiálne medzi šachistami delí na 3 etapy – otvorenie, stredná hra a koncovka. Toto rozdelenie sa však nedá jednoducho popísať a ani neexistuje norma, ktoré by toto rozdelenie popisovala. Počas histórie hrania šachu však šachisti došli k záveru, že niektoré prvé ťahy partie dosahujú lepšie výsledky ako iné. Táto fáza prvých známych ťahov sa dá popísať ako otvorenie. Veľká časť partií sa začína práve týmito otvoreniami, ktorých zoznam sa dá nájsť tu [9]. Po identifikácií otvorenia sa dá vo veľa partiách určiť nasledujúce ťahy bieleho či čierneho.

- **Vylúčenie predikcií podľa hrateľnosti ťahu**. Táto technika spočíva vo využití pravidiel šachu na odstránenie predikcií ťahov, ktoré nemôžu byť v danej pozícií zahrané. Systém by si mohol ukladať súčasnú pozíciu z predošlej postupnosti ťahov a na základe toho rozhodovať, či rozpoznaný ťah aj môže byť zahraný. Napríklad ak by predikcia ťahu bieleho hráča bol ťah jazdcom, avšak biely hráč už jazdca nemá, tak by systém bol schopný túto predikciu vylúčiť.

V súčasnom stave je riešenie prispôsobené na návrh partiáru piešťanského šachového klubu. Ako je spomenuté v sekcií 2.1, každý klub má svoj vlastný návrh partiáru, ktoré sa od seba graficky líšia, čo môže spôsobiť, že postup získania hlavnej tabuľky nebude pre iné návrhy fungovať. Na vyriešenie tohto problému sa zdá byť ako najlepšie riešenie manuálne označenie hlavnej tabuľky pri nahratí partiáru. Aj keď tento spôsob vytvára viacej manuálnej práce pre človeka, rozpoznanie hlavnej tabuľky je kritickým krokom, bez ktorého by bolo nájdenie valídnych buniek veľmi obtiažne.

Obvykle sú pre jednu šachovú hru vygenerované dvoje partiáre. Pre tieto prípady je v budúcnosti možné vytvoriť možnosť nahratia dvojice partiárov miesto jedného a pomocou tejto dvojice zlepšiť presnosť výsledkov. Jedna z možností využitia dvojice partiárov je porovnávnanie koeficientu istoty dvojice predikcií toho istého ťahu a vo výslednej transkripcií sa využije ťah, ktorý ma tento koeficient vyšší.

Ďalšou etapou môže byť pridanie šachovnice na pozeranie súčasnej pozície do užívateľského rozhrania. Systém v súčasnej podobe pracuje na základe práce s textom. Vizuálna ukážka súčasnej pozície na šachovnici môže pomôcť hráčom lepšie sa zorientovať v ktorej časti hry sa nachádzáju a teda predpovedať ďalší ťah, aj keď je ťah napísaný v partiári nečitateľný.

[1] `https://drive.google.com/file/d/1u-RCdbmna1m68gKZzrTlhgo_kp6dTl0l/view?usp=sharing`

Kapitola 8

Záver

Cieľom tejto práce bolo vytvoriť systém, ktorý bude schopný zo snímku šachového partiáru extrahovať sekvenciu ťahov danej hry a vo vhodnom formáte ju exportovať. Vypracovanie systému, ktorý by bol tohto prepisu schopný, by prinieslo pre hráčov šachu množstvo benefitov, pričom hlavný z nich je zredukovanie počtu hodín strávených manuálnym prepisom. Problém digitalizácie šachových partiárov je málo preskúmanou oblasťou a existuje minimálne množstvo systémov, ktoré vedia tento problém riešiť.

Vybrané postupy na analýzu šachových tabuliek využívajú morfologické operácie na zvýraznenie ohraničujúcich čiar a ich následné spracovanie. Podarilo sa vytvoriť systém, ktorý je schopný analyzovať šachový partiár a segmentovať jednotlivé bunky s textom. Rozpoznanie textu pri použití nástroja `pytesseract` dosahuje úspešnosti rozpoznania znakov v texte 16,23 %, pri vstupoch najlepšej kvality, čo znamená, že užívateľ musí opraviť viac ako 80 % ťahov. Po manuálnych opravách je systém schopný exportovať danú hru v štandardnom šachovom formáte PGN. Bolo vytvorené jednoduché uživateľské rozhranie, pomocou ktorého je užívateľ schopný tieto opravy vykonať.

Na zvýšenie presnosti rozpoznania tabuliek bol využitý postup založený na nájdený rotačného uhla pomocou horizontálnych čiar tabuľky. Každá bunka tabuľky je spracovaná tak, aby boli odstránené jej ohraničujúce čiary a takisto aj iné rušivé elementy, ktoré môžu vzniknúť presahom textu z inej bunky.

Možnosti budúceho vývoja sa odrážajú od natrénovania modelu na rozpoznanie šachových ťahov za použitia dátovej sady týchto snímkov. Pomocou vytvoreného systému je možné túto dátovú sadu vytvoriť počas digitalizácie jednotlivých hier.

Hoci vyvinutý systém ešte nie je pripravený na praktické využitie, táto práca poskytuje základy pre budúci vývoj. Adresovaním identifikovaných výziev a nadviazaním na získané poznatky je možné v budúcnosti vyvinúť presnejší systém na digitalizáciu šachových partiárov.

Literatúra

[1] *Chesscom-reaches-100-million-members* [online]. 2022 [cit. 2023-04-17]. Dostupné z: https://www.chess.com/article/view/chesscom-reaches-100-million-members.

[2] Rafael C. Gonzales, R. E. W. *Digital image processing.* 4. vyd. Pearson Education, 2018. ISBN 978-0-13-335672-4. Dostupné z: https://dl.icdst.org/pdfs/files4/01c56e081202b62bd7d3b4f8545775fb.pdf.

[3] Mehul, G., Ankita, P., Namrata, D., Rahul, G. a Sheth, S. Text-based Image Segmentation Methodology. *Procedia Technology.* 2014, zv. 14, s. 465–472. DOI: https://doi.org/10.1016/j.protcy.2014.08.059. ISSN 2212-0173. 2nd International Conference on Innovations in Automation and Mechatronics Engineering, ICIAME 2014. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2212017314000954.

[4] Nam, E. *Understanding the Levenshtein Distance Equation for Beginners* [online]. [cit. 2023-05-01]. Dostupné z: https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0.

[5] Ma, C., Lin, W., Sun, L. a Huo, Q. Robust Table Detection and Structure Recognition from Heterogeneous Document Images. 2022, [cit. 2023-04-22]. DOI: 10.1016/j.patcog.2022.109006.

[6] *Forsyth-Edwards Notation (FEN)* [online]. [cit. 2023-04-26]. Dostupné z: https://www.chess.com/terms/fen-chess.

[7] *Tkinter — Python interface to Tcl/Tk* [online]. [cit. 2023-04-26]. Dostupné z: https://docs.python.org/3/library/tkinter.html.

[8] *Contours : Getting started* [online]. [cit. 2023-04-28]. Dostupné z: https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html.

[9] *Chess openings list* [online]. [cit. 2023-05-01]. Dostupné z: https://chessfox.com/chess-openings-list/.

[10] *Algebraic notation (chess)* [online]. [cit. 2023-05-01]. Dostupné z: https://en.wikipedia.org/wiki/Algebraic_notation_(chess).

[11] *Laws Of Chess* [online]. [cit. 2023-04-17]. Dostupné z: https://www.fide.com/FIDE/handbook/LawsOfChess.pdf.

[12] Ahle, T. [online]. [cit. 2023-04-19]. Dostupné z: https://chess.stackexchange.com/questions/2506/what-is-the-average-length-of-a-game-of-chess.

[13] Eicher, O., Farmer, D., Li, Y. a Majid, N. Handwritten Chess Scoresheet Recognition Using a Convolutional BiLSTM Network. In: Barney Smith, E. H. a Pal, U., ed. *Document Analysis and Recognition – ICDAR 2021 Workshops.* Cham: Springer International Publishing, 2021, s. 245–259. ISBN 978-3-030-86198-8. Dostupné z: https://doi.org/10.1007/978-3-030-86198-8_18.

[14] University of Oslo. *Digital images and image formats* [online]. [cit. 2023-07-05]. Dostupné z: https://www.uio.no/studier/emner/matnat/math/MAT-INF1100/h08/kompendiet/images.pdf.

[15] *The Most Colorful Birds From Around The World* [online]. [cit. 2023-05-07]. Dostupné z: https://www.worldatlas.com/articles/the-most-colorful-birds-of-the-world-where-to-see-them.html.

[16] academic, H. coding. Otsu thresholding - image binarization. [online]. [cit. 2023-03-20]. Dostupné z: https://hbyacademic.medium.com/otsu-thresholding-4337710dc519.

[17] Sauvola, J. a Pietikäinen, M. Adaptive document image binarization. *Pattern Recognition.* 2000, zv. 33, č. 2, s. 225–236. DOI: https://doi.org/10.1016/S0031-3203(99)00055-2. ISSN 0031-3203. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0031320399000552.

[18] Oakley, B. *Python tkinter text modified callback* [online]. [cit. 2023-05-04]. Dostupné z: https://stackoverflow.com/questions/40617515/python-tkinter-text-modified-callback.