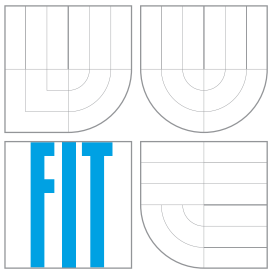
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ROZPOZNÁVÁNÍ KÓDŮ CAPTCHA

CAPTCHA RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

RADEK PAZDERKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MAREK ŽÁK

BRNO 2016

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Pazderka Radek**
Obor: Informační technologie
Téma: **Rozpoznávání kódů captcha**
Captcha Recognition
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se se způsoby vyhledávání textu v obrázku a s dostupnými implementacemi pro jeho přečtení.
2. Shromažděte datovou sadu (captcha obrázky typu black overlap) pro průběžné testování vyvíjeného systému pro rozpoznávání kódů captcha.
3. Na základě získanách poznatků navrhněte a implementujte systém, který dokáže ze zadané webové stránky stáhnout bezpečnostní kódy captcha a provést jejich rozpoznání.
4. Vyhodnoťte výsledky systému na vzorku dat s alespoň 100 obrázky a na webové stránce.
5. Vytvořte stručný plakát prezentující práci, její cíle a dosažené výsledky.

Literatura:

- Šonka M., Hlaváč V.: Počítačové vidění, Grada, 1992, ISBN 8085424673

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Žák Marek, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L. S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zaměřuje na návrh a implementaci aplikace sloužící k rozpoznávání CAPTCHA kódů. Dále se věnuje popisu různých typů CAPTCHA kódů, jejich bezpečnostních prvků a existujících řešení, kterých se v dnešní době využívá k rozpoznání CAPTCHA kódů. Hlavním smyslem práce je otestování bezpečnosti daného typu textového CAPTCHA kódu, který používají webové stránky při obraně proti nelegálním programům.

Abstract

This bachelor thesis is focused on design and implementation of application, which would recognize CAPTCHA codes. It also describes various types of CAPTCHA codes, their security properties and existing solutions which are used today for recognizing CAPTCHA codes. Main goal of this thesis is testing security of certain type of text CAPTCHA code, which is used by web sites for protection against illegal applications.

Klíčová slova

CAPTCHA kód, bezpečnost, strojové učení, segmentace, klasifikace

Keywords

CAPTCHA code, security, machine learning, segmentation, classification

Citace

PAZDERKA, Radek. *Rozpoznávání kódů captcha.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Žák Marek.

Rozpoznávání kódů captcha

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Marka Žáka Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Radek Pazderka
15. května 2016

Poděkování

Rád bych poděkoval mému vedoucímu panu Ing. Marku Žákovi za jeho odbornou pomoc při tvorbě této bakalářské práce.

Obsah

Úvod

Každý, kdo chce v dnešní době vytvořit webovou stránku, buď pro sebe nebo pro někoho na zakázku, měl by dbát na její bezpečnost. Hrozbu představují nelegální programy, které se na webové stránce vydávají za běžného uživatele, ale na rozdíl od něho odesílají větší počet a libovolný typ požadavků na webovou stránku. Webová stránka zpravidla slepě odpovídá na každý požadavek, který jí přijde a nezjistí, že se nejedná o člověka, ale o nelegální program. Webový útočník, který vytvoří tento program, se zajímá pouze o profit nebo získání důležitých informací o systému a tím celou webovou stránku může znehodnotit.

Jako obrana proti takovýmto IT zločincům vznikla CAPTCHA [7]. Hlavní myšlenka CAPTCHA kódů byla rozlišit, zda se jedná o reálného uživatele nebo o program, který se za reálného uživatele pouze vydává. Ochrana pomocí CAPTCHA kódů spočívá v tom, že je vyžadováno po uživateli opsání deformovaného textu nebo např. odpověď na jednoduchou otázku popř. identifikace předmětu v obrázcích. Na tento „triviální“ dotaz dokáže uživatel bez problémů odpovědět, ale pro program je to takřka neřešitelný problém. S postupným zdokonalováním umělé inteligence se programům stále zvyšuje úspěšnost v rozpoznávání bezpečnostního prvku CAPTCHA. Proto je důležité dbát na jeho bezpečnost.

V této bakalářské práci se zaměříme na způsob prolomení jednoduché CAPTCHA ochrany, která se dnes může objevit i na některých méně zabezpečených webových stránkách. V první kapitole si uvedeme základní informace o kódech CAPTCHA. Zavedeme si jeho druhy a bezpečnostní prvky, které by po přidání do CAPTCHA kódu tvořily robotům problém při rozpoznávání. Ve druhé kapitole si popíšeme konkrétně řešený typ CAPTCHA kódu, na který se bude aplikace specializovat. Ve třetí kapitole se budeme věnovat návrhu aplikace, ve které si osvětlíme jednotlivé fáze rozpoznávání. V každé fázi se zaměříme na možné algoritmy a zhodnotíme jejich klady a zápory. Ve čtvrté kapitole si popíšeme celou implementaci navrženého systému. Uvedeme si podrobně implementaci vybraných algoritmů. V páté kapitole se seznámíme s implementovaným grafickým rozšířením bakalářské práce. Kapitola se bude skládat převážně z popisu ukázek grafických prvků aplikace. V páté kapitole otestujeme celou práci. Postavíme ji před potenciální uživatele a před testovací sadu. V závěru zhodnotíme všechny dosažené výsledky, které byly zjištěny v průběhu tvorby celé práce.

Kapitola 1

CAPTCHA kód

V této kapitole se zaměříme na důkladnější popis CAPTCHA kódů. Zejména na jejich druhy, faktory bezpečnosti, na které bychom při výběru neměli zapomenout a možné řešení CAPTCHA kódů neboli jejich prolomení.

1.1 Co je to CAPTCHA kód

CAPTCHA neboli „Completely Automated Public Turing test to tell Computers and Humans Apart“ je Turingův test. Tento test se snaží odlišit člověka a robota na webových stránkách [6]. CAPTCHA kód byl vyvinut v roce 2000 týmem programátorů na Carnegie Mellon University. První CAPTCHA kód vznikl jako forma obrany proti automatickému vkládání URL do katalogu AltaVista. Dříve CAPTCHA kód používal přímé vykreslení textu do obrázku bez jakékoliv deformace písma, které se ale webovým útočníkům velmi jednoduše podařilo překonat. Musely se tedy začít využívat metody, které znesnadnily detekci znaků pro roboty, ale zachovaly čitelnost pro lidi. Mezi tyto metody patří např. deformace textu nebo částečné překrytí jednotlivých znaků. CAPTCHA kód může představovat problém pro zrakově postižené, který lze zmírnit využitím audio CAPTCHA, který kód přehraje. Čím rychleji se vyvíjí software, tím rychleji se musí měnit také CAPTCHA kód. Také z důvodu zhoršující se schopnosti lidí dešifrovat deformované texty se objevuje mnoho nápadů či řešení, jak CAPTCHA vylepšit [12]. Je tedy možné, že se místo klasického textu setkáme s rozpoznáním zvířete na obrázku, s 3D otáčením obrázků či s mini hrami apod. viz Kapitola 1.2.

1.2 Druhy CAPTCHA kódů

V této sekci se seznámíme s vybranými druhy CAPTCHA kódů. Uvedeme si jejich hlavní myšlenku, popř. výhody a nevýhody.

1.2.1 Opisování textu z obrázku

- **Klasický CAPTCHA kód**

  Jedná se o základní CAPTCHA kód. Po uživateli je požadováno opsání libovolně deformovaného textu. Tento typ je v dnešní době stále nejvíce používaný na webových stránkách. Podrobnosti o možnostech zabezpečení tohoto typu CAPTCHA kódu na-

leznete v kapitole Bezpečnost 1.4. Na obrázku 1.1 je uveden základní CAPTCHA kód se vstupním polem, do kterého se přepisuje text z obrázku.

Obrázek 1.1: Základní CAPTCHA kód.

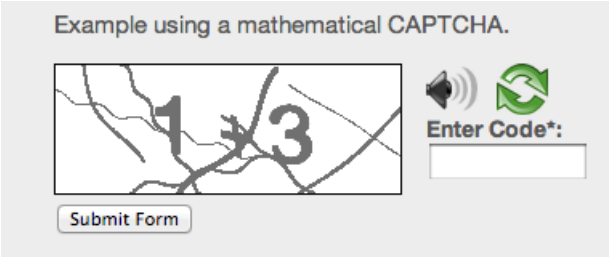
- **reCAPTCHA**

  Tento typ CAPTCHA kódu slouží k pomoci s digitalizací tištěných médií např. knih, časopisů atd. Principem je, že se naskenuje text dokumentu a tento text se předá OCR [4] programu na analýzu. V případě, že OCR program není schopen text s jistotou přečíst, je tento text přidán do CAPTCHA kódu společně s textem, který je známý. Podle přepisu od uživatele se zjišťuje, co vlastně obsahuje neznámé slovo. Tento CAPTCHA kód se používá jako ochrana webových stránek a také jako pomocník při rozpoznávání nečitelných slov tištěných dokumentů. Na obrázku 1.2 je uvedena ukázka reCaptcha kódu.

Obrázek 1.2: ReCaptcha.

1.2.2 Řešení úloh

Tento typ CAPTCHA kódu vyžaduje nejen správné přečtení, ale i správné logické vyřešení příkladu. Většina robotů nemají implementované logické myšlení a schopnost řešení jednoduchých úkolů. Mezi další varianty patří CAPTCHA kód, který vyžaduje odpověď na jednoduchou otázku: „Jaký je čtvrtý den v týdnu?“, „Kolik nohou má pes?“ atd. Obrázek 1.3 demonstruje matematický příklad 1+3, který vyžaduje po uživateli číslo 4.

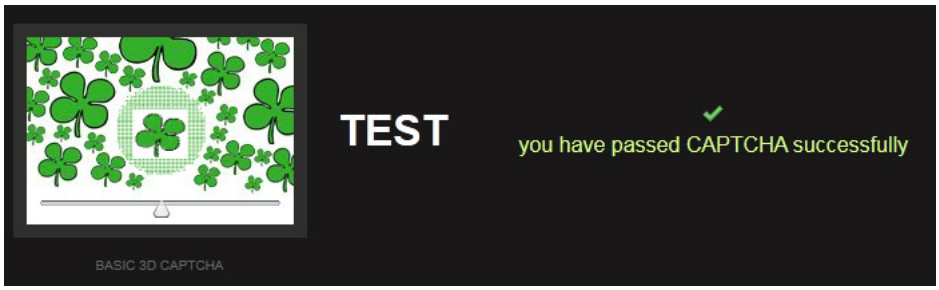


Obrázek 1.3: Matematické řešení CAPTCHA kódu.

1.2.3 3D CAPTCHA kód

Cílem tohoto CAPTCHA kódu je udělat Adobe flash [3] animaci s posuvným ukazatelem. Při pohybu tímto ukazatelem se v obrázku otáčejí trojúhelníky, na které byl obrázek původně rozsekán. Uživatel má za úkol uvést posuvník do takové pozice, aby vznikl obrázek vizuálně podobný svému okolí. Poté klikne a tlačítko pro ověření správnosti a aplikace mu vrátí výsledek – zda se nachází v toleranci nebo ne. Tento inovativní druh CAPTCHA kódu je velice uživatelsky přívětivý. Uživatel si tak může „zahrát“ velice jednoduchou hru a tím ubezpečit webovou stránku, že není robot [17].

Obrovská výhoda tohoto druhu CAPTCHA kódu spočívá v tom, že se může použít libovolný obrázek jako předloha. Roboti si nebudou moci zaznamenat všechny možné kombinace obrázků a tím se zvýší bezpečnost tohoto CAPTCHA kódu. A také většina robotů není vybavena mechanizmem na stahování a následné spouštění Adobe flash aplikací z webových stránek. Tato vlastnost většinou útočníka odradí. Na druhou stranu od Adobe flash se postupně odstupuje a nahrazuje se HTML5 a tak se časem může stát nepodporovaným na webových prohlížečích. Na obrázku 1.4 je uvedena ukázka 3D CAPTCHA kódu.



Obrázek 1.4: 3D CAPTCHA kód.

1.2.4 Rozdělení CAPTCHA kódu pomocí CSS stylu

Další možností, jak předejít přečtení obrázku robotem a zachovat jeho čitelnost je rozdělení obrázku na jednotlivé části a jeho znovu-složení pomocí CSS stylu. Obrázek se rozdělí

na dvě až tři části a jednotlivé části se vloží do odlišných částí HTML kódu. Znovu se složí až pro uživatele. Robot tak obdrží pouze jednotlivé části obrázku, z kterých nic nevyčte, protože nejsou úplné. Tento obrázek se navíc může podbarvit jako zbytek podkladu stránky, takže bude zapadat do designu. Navíc font i velikost písma budou pro lidského uživatele čitelné, bez zbytečného přeškrtávání, naklánění nebo jiné deformace písma. CAPTCHA kód také nebude zabírat tolik prostoru na webové stránce, který se poté může využít pro další sdělování potřebnějších informací pro uživatele [23]. Na obrázku 1.5 je uvedeno možné rozdělení daného CAPTCHA kódu.

Obrázek 1.5: CAPTCHA kód založený na CSS a HTML.

1.3 Existující řešení CAPTCHA kódů

- **Optical Character Recognition**

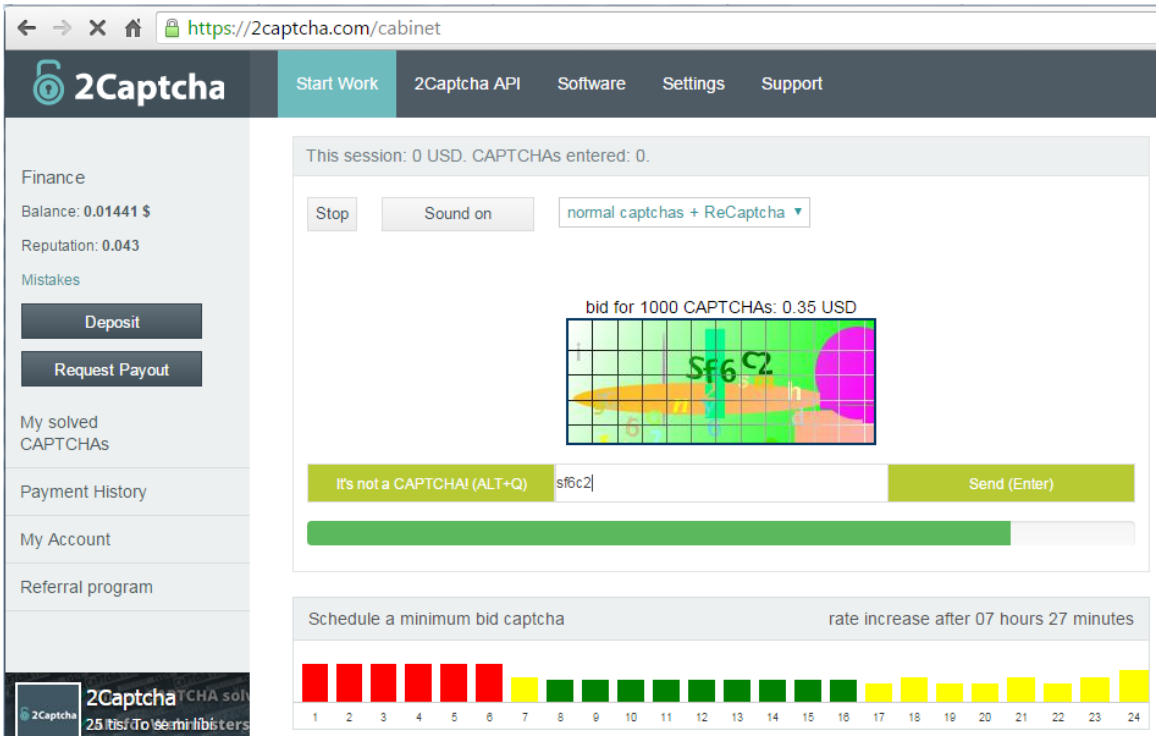
  Optical Character Recognition (OCR) slouží k převodu skenovaného textu do digitální podoby. Na základě této technologie vzniklo několik programů, které řeší CAPCHA kódy. Jeden z nejlepších uváděných placených programů pro řešení CAPTCHA kódů je GSA Captcha Breaker, který umožňuje rozpoznání přes 600 typů kódů [2].

- **Řešení založené na levné lidské práci**

  Existuje řada webových stránek, které poskytují řešení CAPTCHA kódů za určitý obnos peněz [1]. Cena se v dnešní době pohybuje okolo dvou dolarů za správné vyřešení 1000 kódů CAPTCHA. Většina společností umožňuje i registraci a možnost opisování CAPTCHA kódů. Za správné opsání přibývají na účet malé obnosy peněz. Při opsání dostatku CAPTCHA kódů si uživatel může hotovost vybrat nebo využít služby společnosti pro opsání vlastních CAPTCHA kódů.

  Výhodou této služby je nízká cena za vyřešení CAPTCHA kódů. Na druhou stranu zákazník musí počítat s faktem, že se jeho CAPTCHA kód bude rozpoznávat v řádech jednotek sekund.

  Této služby využívali programy, které stahovali videa z určitých serverů a každé video bylo chráněno CAPTCHA kódem. Valnou většinu zákazníků tvoří lidé, kteří se nechtějí zdržovat s opisováním CAPTCHA kódů nebo weboví útočníci, kteří útočí na webové stránky chráněné velice silným CAPTCHA kódem. Na obrázku 1.6 je uvedena ukázka webové stránky zabývající se řešením CAPTCHA kódů za peníze.

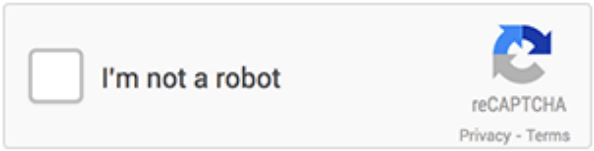


Obrázek 1.6: Řešení CAPTCHA kódu založený na lidské práci.

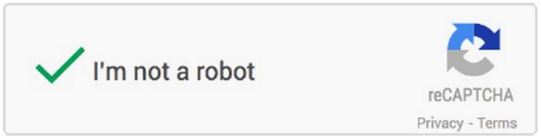
- **reCaptcha od firmy Google prolomená vlastní silou**

  Služba reCaptcha od firmy Google je široce používaná CAPTCHA ochrana využívána mnohými populárními webovými stránkami proti automatickým robotům provádějící zločinné aktivity. Když uživatel navštíví webovou stránku chráněnou od reCaptcha, uvidí jeden bílé zaškrtávací pole, s textem „I'm not a robot" obrázek 1.7. Při odkliknutí zaškrtávacího políčka se spustí JavaScript, který sbíral informace o chování uživatele. Pokud prohlásí, že se jedná na 100 % o uživatele, tak hned povolí přístup a uživatel může pokračovat viz obrázek 1.8. V případě, že JavaScript nemá dost informací o uživatelově aktivitě, tak požádá o uživatele o identifikaci devíti obrázků, viz obrázek 1.9.

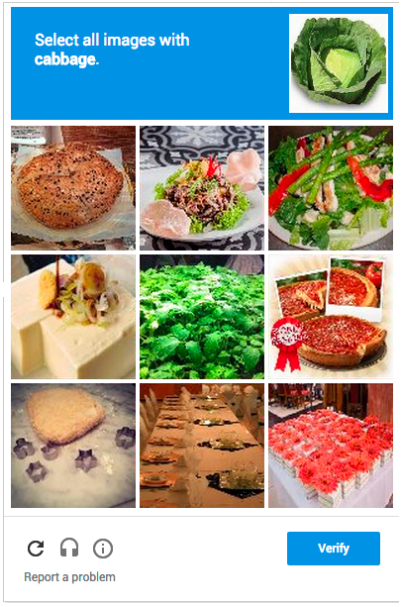
  Prolomení této ochrany spočívá v tom, že se jednotlivé obrázky vyhledají pomocí *images google* a podle nalezených klíčových slov, názvů dvaceti nejlepších příspěvků a dalších informací o obrázku se dá s velkou úspěšností odhadnout, co se na daném obrázku nachází [20].



Obrázek 1.7: reCaptcha od firmy Google 1.



Obrázek 1.8: reCaptcha od firmy Google 2.



Obrázek 1.9: reCaptcha od firmy Google 3.

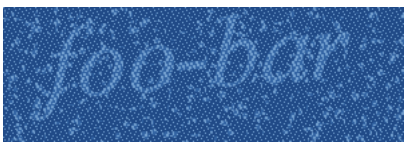
1.4 Bezpečnost textových CAPTCHA kódů

CAPTCHA kód se označuje za prolomený, když existuje program, který dokáže správně přečíst alespoň 1 % kódů [5]. Nejobtížnější fází rozpoznání pro robota je segmentace jednotlivých písmen. Proto je velice žádoucí na tento fakt brát zřetel a ve výsledném CAPTCHA kódu co nejvíce znemožnit robotům segmentaci písmen. Ale pouze do takové míry, aby byl stále CATPCHA kód čitelný pro člověka.

Při výběru CAPTCHA kódů bychom měli dbát na určité parametry, které sníží množství prolomení od útočníků:

- **Pozadí textu**

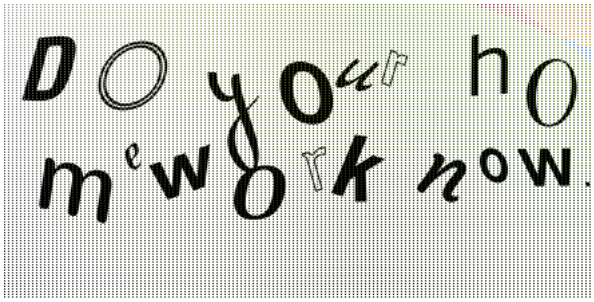
  Pozadí textu by mělo být velice podobné jako text a proměnlivé na CAPTCHA kódech daného typu. Nevhodné je statické pozadí, které lze jednoduchým způsobem odstranit. Na obrázku 1.10 je příklad dobře zvoleného pozadí textu.



Obrázek 1.10: Ideální pozadí textu.

- **Množina písmen**

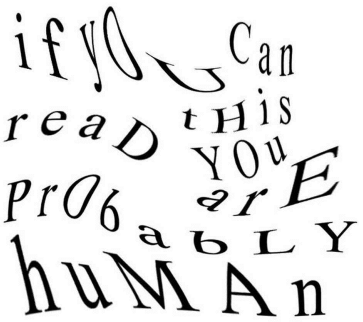
  Je velice žádoucí využít co nejvíce znaků, popř. i speciální znaky. Lehce prolomitelné CAPTCHA kódy většinou neobsahují všechna písmena abecedy. Například obsahují pouze velká písmena a jen část abecedy. Některé mají i redukovanou řadu číslic. V případě nedostačujícího počtu typů znaků se velice usnadní prolomení daného CAPTCHA kódu. V ideálním případě CAPTCHA kód obsahuje všechna písmena a číslice různých typů. Na obrázku 1.11 je ukázka přijatelné množiny písmen. Každé písmeno je napsáno jiným stylem, což je velice vhodné.



Obrázek 1.11: Množina písmen.

- **Deformace**

  Pro ztížení práce robota je dobré v rozumné míře deformovat text. Jak jeho hrany, tak i celý tvar takovým způsobem, aby byl výsledný znak co nejvíce odlišný od písmen ostatních typů CAPTCHA kódů. Na obrázku 1.12 je uvedena vhodná deformace textu.



Obrázek 1.12: Deformace písmen.

- **Náhodné pootočení**

  Dalším faktorem je rotace písmen. Tento bezpečnostní faktor není až tak zásadní, protože jednoduchým algoritmem lze vyseparovaná písmena a číslice otočit zpět. Algoritmus vychází z toho, že alfanumerické znaky jsou správně postaveny v případě, že mají nejmenší možnou šířku.

- **Barva a velikost**

  Při změně barvy a její intenzity každého znaku se znemožní robotům jednoduchého převodu každého barevného znaku na černý a pozadí ponechat bílé. Zavedením libovolné velikosti jednotlivých písmen se také zvýší ochrana daného CAPTCHA kódu.

Robot pak musí daného písmena modifikovat velikost a tím ztrácí úspěšnost celého rozpoznání. Obrázku 1.13 reprezentuje měnící se velikost, barevnost a pootočení písmen.

Obrázek 1.13: Barva, velikost a pootočení písma.

- **Překrývající se písmena**

  Překrývání písmen je velice důležitý ochranný prvek každého textového CAPTCHA kódu. Znemožňuje se jednoduchá segmentace písmen. A jak je již známo, nejtěžší práci mají roboti se segmentací písmen. Na obrázku 1.14 je demonstrována postupně se zlepšující bezpečnost v podobě překrývajících se písmen. Důležité je ale brát zřetel na čitelnost pro člověka. Není vhodné písmena překrýt do takové úrovně, aby se nedala oddělit ani lidským zrakem.

Obrázek 1.14: Překrývající se písmena.

Kapitola 2

Analýza problému

V dnešní době je největším problémem v textových CAPTCHA kódech jejich malá bezpečnost.

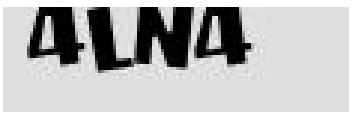
Bakalářská práce se zaměřuje na ochranu webových stránek zabezpečených převážně CAPTCHA kódy typu Black overlap (viz Kapitola 2.1). Hlavní důraz je kladen na správné rozpoznání daného CAPTCHA kódu. Podle výsledku správného rozpoznání se bude moci oznámit, zda je daný CAPTCHA kód lehko prolomitelný. Tato informace by měla být velice ceněná pro vývojáře webových stránek.

V případě, že je rozpoznán jeden CAPTCHA kód ze sta, může se daný typ CAPTCHA kódu označit jako nedostačující pro webovou stránku. Webovým spamovacím robotům bohatě stačí 1% úspěšnost, aby udělaly na webových stránkách i nevratné škody. Podle této hranice se budeme orientovat, zda byla aplikace úspěšná nebo nikoliv.

2.1 Řešení CAPTCHA kódu typu Black overlap

Typ CAPTCHA kódu Black overlap se vyznačuje tím, že všechny kódy jsou v odstínu šedé nebo jsou pouze černobílé a obsahují výrazná písmena, která se dají lehko pro člověka přečíst. Písmena se překrývají pouze v ojedinělých případech. I přes výslovné varování se tento typ CAPTCHA kódu objevuje na mnoha webových stránkách či fórech. Příkladem webových stránek, kteří ještě dnes používají tento typ CAPTCHA kódu: Firma O_2 [15], webový obchod Sladký méďa [19] nebo české populární fórum pro mladou generaci: Zpovědnice [22].

Zaměříme se na CAPTCHA kód z fóra Zpovědnice, který se rozděluje do tří základních skupin. Jedná se o typ s deformovanými hranami, zrcadlově otočenými písmeny a tučně psaným textem. Na první pohled je zřejmé, že se jedná o lehko prolomitelný CAPTCHA kód z důvodu jednoduché segmentace jednotlivých písmen. Na obrázcích 2.1a, 2.1b a 2.1c jsou uvedeny všechny tři typy CAPTCHA kódu.



(a) Tučné písmo.

(b) Zrcadlově otočené písmo.

(c) Deformované hrany.

Obrázek 2.1: Black overlap.

Kapitola 3

Návrh aplikace

V této kapitole se seznámíme s návrhem aplikace, která se bude vytvářet. Jedná se o aplikaci, která se skládá ze tří navzájem nezávislých dílčích programů. Hlavní součást práce by měla být konzolová aplikace, která rozpoznává daný CAPTCHA kód. Jako součást hlavní aplikace by měl být podpůrný skript, který má za úkol stahovat CAPTCHA kódy z libovolné webové stránky. Rozšíření by mělo být v podobě grafického uživatelského rozhraní (GUI) umožňující uživatelsky přívětivější práci s učením dané datové sady a následné rozpoznání.

3.1 Aplikace pro řešení CAPTCHA kódů

Aplikace pro řešení CAPTCHA kódů by měla být napsána v programovacím jazyce C++. C++ je ideální volbou, protože se jedná o kompilovaný jazyk a tím pádem i výpočetně rychlejším oproti např. skriptovacím jazykům. Na rychlost je brán velký zřetel, protože se v této aplikaci bude provádět většina výpočtů a potenciálního uživatele aplikace by odrazovalo dlouhé čekání na vyřešení zadaných CAPTCHA kódů.

Při návrhu aplikace je kladen velký důraz na rychlost, optimalizovanost a znovupoužitelnost kódu. Aplikace by se měla členit do fází, ve kterých každá fáze přijímá určitý typ informací na vstupu a na výstupu vrací jiný typ, který si převezme další fáze. Není možné, aby dvě fáze pracovaly s jedním typem informací. To by způsobilo nečitelnost a neoptimálnost kódu.

Navrhované fáze rozpoznávání:

- **Předzpracování**

  VSTUP: Bitmapa (CAPTCHA kód)

  VÝSTUP: Binarizovaná bitmapa

  Tato fáze by měla mít za úkol „vyčistit“ možné rušivé pozadí obrázku, odstranit nežádoucí osamocené kousky písmen, popř. převést barevný CAPTCHA kód do odstínu šedé. Tento převod je velice důležitý, protože následná funkce prahování vyžaduje na vstupu obrázek v odstínu šedé. Při použití prahování se ztrácí informace o intenzitě šedé v obrázku. Tato negativní vlastnost prahování nám nepřináší žádné omezení, či ztrátu důležitých informací, protože je cílem této fáze předat pouze barvy černé nebo bílé. Proto je pro náš případ velice vhodnou volbou.

- **Segmentace**

  VSTUP: Binarizovaná bitmapa

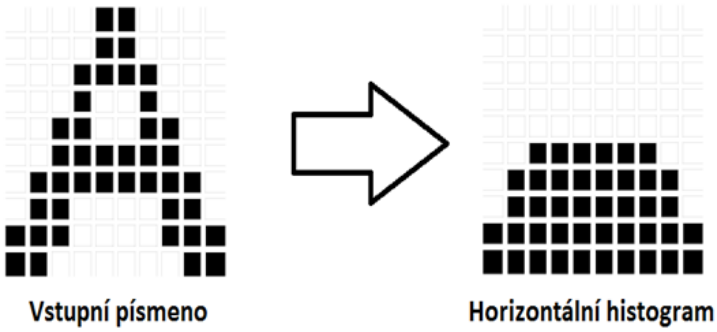
  VÝSTUP: Histogramy jednotlivých písmen

  Do této fáze vstupuje již „očištěný“ obrázek. Cílem je rozdělit daný obrázek na písmena / tvary a z těchto rozdělených písmen / tvarů získat horizontální histogram a předat ho do další fáze. Pomocí horizontálního histogramu se budou identifikovat a porovnávat všechna písmena v následující fázi. Na obrázku 3.1 je demonstrovaný převod rastrového písmene na horizontální histogram.

  Existuje řada algoritmů pro segmentaci písmen z obrázku. První možností je využití horizontálního histogramu. Algoritmus spočívá v tom, že zjistíme, jaké sloupce jsou prázdné, a tím můžeme odhadnout, kde písmeno začíná a kde končí. Výhodou algoritmu je jeho vysoká rychlost segmentace. Nevýhodou je, že je tento algoritmus využitelný pouze v případě, že se písmena nepřekrývají.

  Druhou možností je využití konvoluce [8] pro detekci hran. Pomocí dobře zvoleného kernelu[1] můžeme detekovat hrany všech písmen a tím je od sebe oddělit. Tento algoritmus se převážně využívá v neuronových sítích CNN (Convolutional Neural Network).

  Třetí možností je semínkové vyplnění [14]. Metoda spočívá v tom, že se nalezne první pixel písmene a od tohoto pixelu se začnou všechny jeho sousedé stejné barvy přenášet do nového obrázku, čímž se oddělí jedno písmeno od ostatních. Varianty tohoto algoritmu se liší podle výběru sousedních pixelů. Zda se vybírají pouze čtyři okolní pixely nebo zda se vybírá všech osm sousedů. Tento algoritmus se často používá z důvodu jednoduché implementace a použitelnosti na libovolných obrázcích pro segmentaci všech jejich tvarů. Z tohoto důvodu se tento algoritmus použije v implementaci.



Obrázek 3.1: Převod písmena na horizontální histogram.

- **Učení se datové sady**

  VSTUP 1: Histogramy písmen

  VSTUP 2: Informace o textu v CAPTCHA kódu

  VÝSTUP: Uložení informací do souboru

  Tato fáze by měla dostat sadu histogramů z fáze segmentace a u každého histogramu by měla obdržet informaci, které písmeno s tímto histogramem koresponduje. Tyto informace by si měla uložit do souboru.

[1] Matice, která se aplikuje na celý obrázek

- **Načtení datové sady**

  VSTUP: -

  VÝSTUP: Načtení informací ze souboru

  Tato fáze by měla mít na starost načítání uložených informací ze souboru. Tyto informace by měla umět uložit do vnitřní struktury programu pro rychlý přístup k datům.

- **Klasifikace**

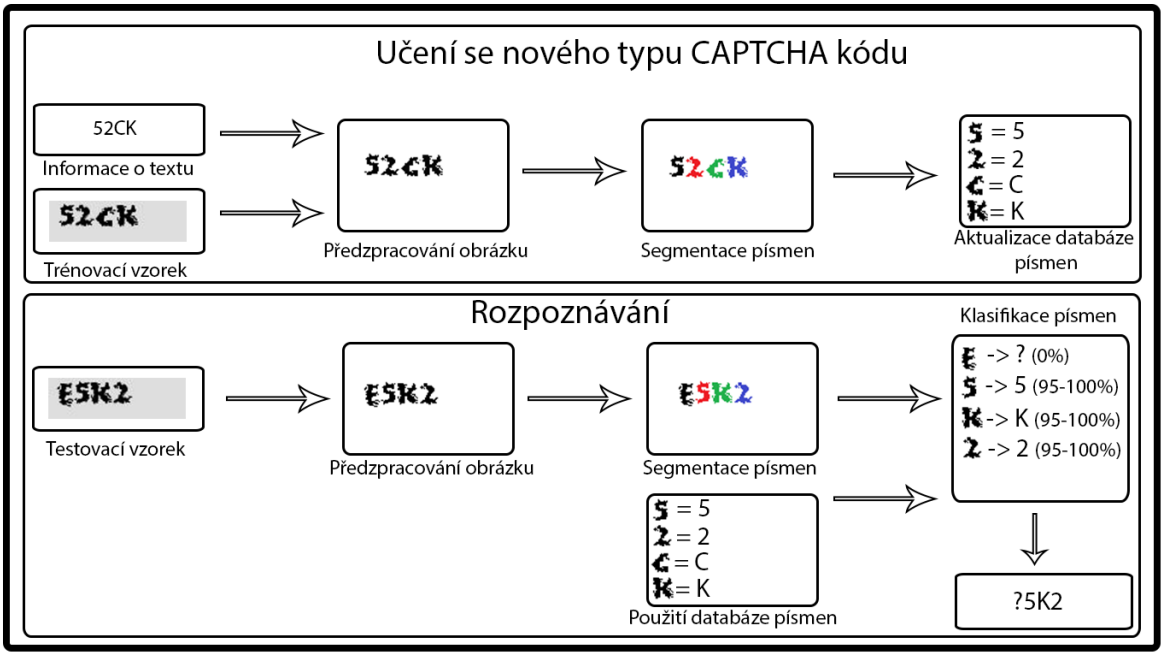
  VSTUP 1: Načtené informace ze souboru

  VSTUP 2: Histogramy jednotlivých písmen

  VÝSTUP: Text s výsledkem klasifikace

  Cílem této fáze měl být systém, který dokáže převzít naučená data původně ze souboru a histogramy z fáze segmentace a porovnat každý histogram s každým a provést vyhodnocení. Dvojici s nejlepší procentuální shodu by měl vytisknout na obrazovku. Kromě této metody existují pro klasifikaci další algoritmy. Jedná se např. o algoritmus K-Nearest Neighbors (KNN), který vybírá k nejbližších sousedů [13].

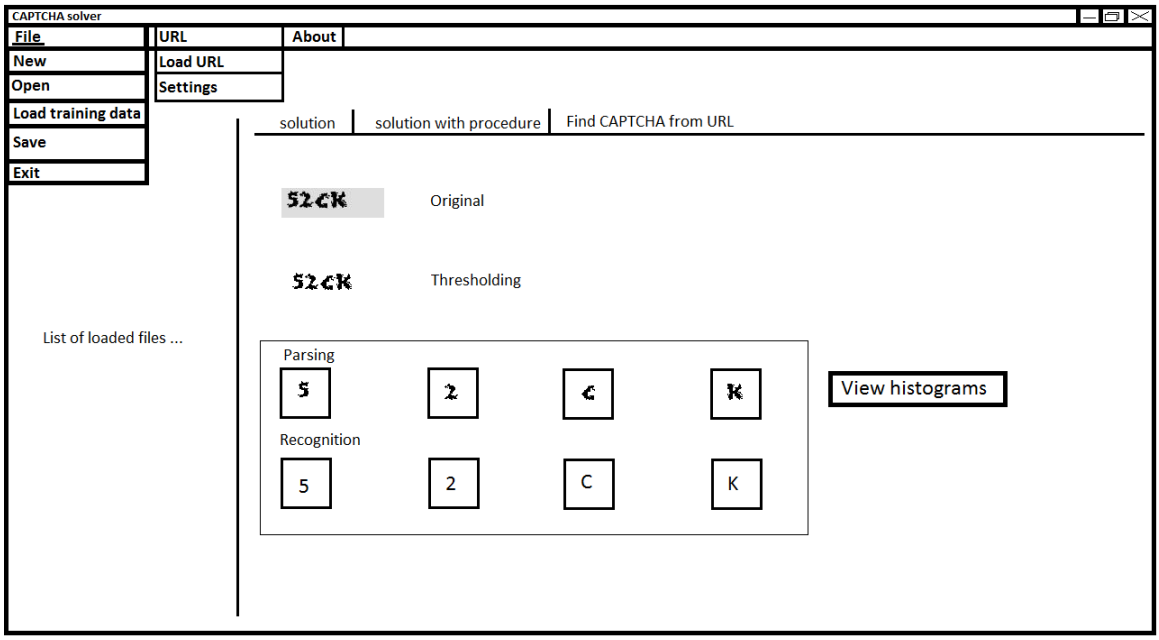
Na obrázku 3.2 lze vidět přehledný souhrn jednotlivých navrhovaných fází. Obrázek také zobrazuje učení se nových znaků a následné rozpoznání. Programu se předal pouze jeden obrázek pro učení. Obsahoval znaky „5“, „2“, „C“, „K“. Při testování dostal program znaky „E“, „5“, „K“, „2“. Všechny znaky kromě znaku „E“ dokázal rozpoznat, protože se je naučil. Při trénování je potřeba mít vybrané takové CATPCHA kódy, které obsahují všechny znaky obsažené v daném typu CAPTCHA kódu.



Obrázek 3.2: Mockup výpočetního jádra.

3.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní bude napsané pomocí knihovny Qt. Qt je velice rozšířená multiplatformní knihovna pro vytváření grafických aplikací. Je to knihovna programovacího jazyka C++, což nám velice vyhovuje. Cílem grafického uživatelského rozhraní bude vytvoření prostředí poskytující uložení si informací o libovolném typu CAPTCHA kódu a následné použití naučené sady k rozpoznání. Dále bude umožňovat zobrazení každé fáze navržené výše (Kapitola 3) pro studijní účely. Toto zobrazení postupu rozpoznávání bude navrženo pro libovolný typ CAPTCHA kódu obsahující libovolný počet znaků. A v poslední řadě zobrazení stažených obrázků popř. CAPTCHA kódů z dané webové stránky.



Obrázek 3.3: Mockup GUI.

3.3 Stahování obrázků z webových stránek

Stahování obrázků z webových stránek bude mít na starost skript, který převezme URL, formáty stahovaných obrázků, minimální a maximální povolenou velikost a další podobné filtry na stahované obrázky. Všechny stažené obrázky by si měl uložit do dočasné složky a jako výstup by měl vrátit cesty ke všem staženým obrázkům. Tento skript bude napsaný ve skriptovacím jazyce Python verze 3.4. Výběr jazyka Python je důvodu jednoduché implementace HTTP a HTTPS komunikace.

Kapitola 4

Popis implementace

V této kapitole se seznámíme s podrobným popisem implementace celé bakalářské aplikace bez rozšíření, které bude popsáno až v Kapitole 5.

4.1 Aplikace pro řešení CAPTCHA kódů

Rozpoznávání CAPTCHA kódu je rozděleno do pěti fází:

4.1.1 Předzpracování

Tato fáze na vstupu obdrží bitmapu, kterou musí umět zpracovat (popsáno níže). Fáze dále využívá funkci prahování [9], která pracuje pouze s obrázkem v odstínu šedé. Intenzita prahu je nastavena na takovou hodnotu, aby byl co nejvíce zachován tvar písmene a zároveň se odstranil nežádoucí šum z pozadí textu. Intenzita prahu je nastavena na 30 %, což odpovídá hodnotě 75. Je to z důvodu toho, že jsou písmena / tvary výrazně tmavé a hodnota 75 ideálně předpřipraví daný řešený CAPTCHA kód. V této fázi je také implementovaná metoda semínkového vyplnění. Je tu z důvodu toho, aby označila velké tvary – písmena a malé osamocené celky odstranila. Každé nalezené písmeno označí jinou barvou. Je to z důvodu optimalizace. Aby následná fáze segmentace nemusela implementovat stejný algoritmus.

Popis použitých metod:

- **Zpracování formátu bitmapa**

  Aplikace dokáže přečíst pouze CAPTCHA kódy uložené v rastrovém formátu bitmapa. Celý obrázek je popsán pixely. Každý pixel se skládá ze tří barevných složek RGB.

  - *Hlavička bitmapy*

    Hlavička bitmapy má vždy velikost 54B. Každý byte hlavičky informuje o jiné vlastnosti bitmapy. Některé byty mohou být nulové a na výsledné bitmapě to ve finále nic nezmění. Celá hlavička bitmapy je zapsána stylem Big-endian[1]. Obrázek 4.1 demonstruje přehlednou tabulku povinných dat v hlavičce bitmapy. Umístění reprezentuje index, na kterým se nachází požadovaná data.

[1] Jedná se o způsob uložení dat v paměti. Nejvyšší řád čísla je uložen nejvíce vlevo.

Tabulka 4.1: Hlavička bitmapy - povinné údaje.

Umístění	0-1	2-5	10	14	18-21	22-25	26	28	38-39	42-43
Velikost	2B	4B	1B	1B	4B	4B	1B	1B	2B	2B
Hodnota (popis)	0x424D	Celková velikost	0x36	0x28	šířka	výška	0x01	0x18	0xC40E	0xC40E

Výpočet paddingu

Každý řádek bitmapy musí být zarovnaný na násobky čtyř. Tento výpočet je důležitý pro správné uvedení velikosti celé bitmapy do hlavičky. Pro výpočet hodnoty padding a následně velikosti celé bitmapy poslouží tento fragment kódu 4.1.

```
int nLineBytes = nWidth * 3;
while ((nLineBytes + (*padding)) % 4 != 0)
{
        ++(*padding);
}
nSizeOfFile = nLineBytes + (*padding);
nSizeOfFile *= nHeight;
nSizeOfFile += HEADER_SIZE;
```

Kód 4.1: C++ code - Bitmap padding.

V proměnné *nLineBytes* je uložen počet bytů na každý řádek bez bytů pro padding. Ve smyčce while se vypočítá hodnota padding. Následně se do proměnné *nSizeOfFile* uloží celkový počet bytů celého bitmap souboru.

– *Data*

Každý byte dat bitmapy reprezentuje jeden RGB kanál jednoho pixelu. Kanály pixelů jsou v bitmapě uspořádány pozpátku. Prvně se v datech nachází modrý, pak zelený a poslední červený kanál. Na konci každého řádku se nachází padding. Podle hodnoty padding se určuje, kolik se vloží nul na konec každého řádku.

- **Převod barevného obrázku do odstínu šedé**

  V případě, že se na vstupu objeví barevný CAPTCHA kód, tak se prvně zavolá převod do GrayScale [18]. Kontrola, zda se jedná o barevný obrázek se provádí porovnáním všech barevných kanálů RGB. Barevný obrázek se detekuje v případě, že se alespoň u jednoho pixelu nerovnají všechny tři jeho barevné složky.

  Algoritmus spočívá v tom, že se váhově ohodnotí každá barevná složka RGB. Nejvíce je hodnocená zelená barva, protože je na ní lidské oko nejvíce citlivé. Poté je barva červená a poslední je modrá. Výsledný pixel se vypočítá podle rovnice 4.1. Y' značí hodnotu, která se vloží do všech barevných složek daného pixelu.

$$Y' = 0.299R' + 0.587G' + 0.114B'. \tag{4.1}$$

- **Prahování**

  Jedná se o základní metodu segmentace obrazu založenou na hodnocení jasu každého pixelu [9]. Základem algoritmu je nalezení hodnoty prahu, pro který bude platit, že všechny pixely jasu nižší než práh budou nastaveny na hodnotu A a vyšší nebo rovný prahu budou nastaveny na hodnotu B. Hodnoty A a B se většinou volí jako 0 a 255, neboli černá a bílá. Rovnice 4.2 popisuje algoritmus prahování.

$$f(c) = \begin{cases} A & \text{pokud } c < prah \\ B & \text{pokud } c \geq prah \end{cases} \tag{4.2}$$

- **Semínkové vyplnění**

  Tento algoritmus se používá pouze v rastrové grafice. V aplikaci tento algoritmus vybere první nalezený černý pixel a začne se okolo něj v osmi-okolí rozrůstat po dalších černých pixelech. V případě, že se algoritmus už nemá kam rozrůstat, vyhodnotí se velikost jeho nalezeného tvaru. Pokud spadá pod hranici 30 pixelů, je označen barvou pro následné odstranění. Pokud se ale jedná o velký nalezený tvar, tak se vygeneruje barva od 1 do 254 a touto barvou se následně přemaluje. Barevný rozsah od 1 do 254 se generuje proto, že hodnota 255 je rezervována pro barvu pozadí a hodnota 0 je rezervována pro zatím neohodnocené tvary. Poté, co algoritmus nenajde již další černé pixely[2] ukončuje svou činnost.

4.1.2 Segmentace

Segmentace je jedna z nejdůležitějších fází celého rozpoznání CAPTCHA kódu, proto je potřeba jí věnovat nejvíce času. V programu segmentace dostává na vstupu jeden RGB kanál bitmapy. V tomto kanálu jsou barevně odlišeny jednotlivá písmena nalezená pomocí semínkového vyplnění z předchozí fáze. Každé takto nalezené písmeno je transformováno do horizontálního histogramu (viz obrázek 3.1) a uloženo do vektoru. V případě, že program zjistí, že semínkové vyplnění identifikovalo dvě písmena namísto jednoho, zavolá funkci pro nalezení nejvhodnějšího řežu písmen (princip bude popsán níže). Kontrola, zda se jedná pouze je jeden znak se provádí podle maximální šířky znaku naučené sady. Nemůže nastat případ, že by tvar s několikanásobnou velikostí představoval pouze jedno písmeno. Podle toho kolikrát je tvar větší kolik písmen se z něj musí vyextrahovat.

Popis použitých metod:

- **Nalezení nejvhodnějšího řezu písmen**

  Funkce dostává na vstupu jeden nebo více histogramů, které jsou podezřelé z toho, že skrývají informaci o dvou nebo více písmenech. Prvně funkce odhadne podle velikosti histogramu, kde by se mohly nacházet řezy. Od tohoto místa si vymezí rozsah, ve kterém se následně bude hledat nejideálnější řez. Jsou vybrány tři nejlepší řezy tzn. sloupce v histogramu, které mají nejmenší hodnotu. Po nalezení všech řezů funkce vytvoří kombinaci všech nalezených řezů a pomocí nich rozdělí původní velký histogram na několik malých. Pro řez dvou písmen vzniknou tři možné varianty, pro rozdělení tří písmen vznikne devět variant, atd. Po skončení této funkce algoritmus bude mít k dispozici všechny varianty histogramů, které předá do poslední fáze klasifikace.

[2] Hodnota pixelu 0

4.1.3 Učení se datové sady

V této fázi dostává aplikace trénovací data. Nezbytnou součástí pro natrénování zadaných písmen / tvarů je sada CAPTCHA kódů. Sada se skládá z několika obrazových souborů s CAPTCHA kódem. Každý soubor musí být uložen ve formátu bitmap (bmp) a pojmenován typem a obsahem CAPTCHA kódu[3]. Aplikace zpracuje každý soubor s CAPTCHA kódem, ze kterého si získá všechna písmena a o každém písmenu si uloží informaci o jeho horizontálním histogramu[4] do souboru. Písmena se nesmí překrývat. V případě, že se písmena překrývají, aplikace prohlásí, že nesedí počet písmen v názvu souboru a počet písmen na obrázku a daný soubor je označen jako nevalidní a je vyloučen z trénovací sady. Při zpracovávání jednotlivých souborů algoritmus prochází dvěma ze tří fází rozpoznávání. Jedná se o předzpracování a segmentaci. Obě tyto fáze jsou popsány výše. Takto naučená data se využívají v poslední fázi rozpoznávání – klasifikace popsaná níže.

Data jsou uložena ve formátu JSON [10]. Proto je zapotřebí využít jsoncpp [11], který dokáže rozparsovat libovolný JSON soubor.

4.1.4 Načtení datové sady

Tato fáze načte naučená data ze souboru a uloží si je do složité vnitřní datové struktury v programu pro rychlejší přístup. Načtená data jsou jako v předchozí fázi uložena ve formátu JSON a pro parsování dat je také použit jsoncpp.

4.1.5 Klasifikace

Tato závěrečná fáze prvně získá informaci o typu CAPTCHA kódu. Následně porovnává naučená data získaného typu s daty převzatými z fáze segmentace a rozhoduje o jaké písmeno / tvar se jedná.

Popis použitých metod:

- **Získání typu CAPTCHA kódu.**

  Klasifikace prvně musí získat informaci, který typ CATPCHA kódu bude rozpoznávat. Tento algoritmus se provádí z důvodu zvýšení rychlosti a úspěšnosti rozpoznání.

  - *Optimalizace.*

    V databázi naučených znaků se může nacházet libovolný počet typů CAPTCHA kódů. Je tedy vhodné projít pouze jedenkrát celou databázi naučených písmen a získat z ní typ CAPTCHA kódu, který obsahuje pouze fragment ze všech uložených znaků místo toho, aby se pro každé rozpoznávané písmeno procházela celá databáze znovu.

  - *Zvýšení úspěšnosti rozpoznání.*

    Úspěšnost rozpoznání se zvýší díky tomu, že algoritmus bude porovnávat menší počet písmen.

  Algoritmus převezme vzorek jednoho písmena z rozpoznávaného CAPTCHA kódu a porovná ho s celou databází naučených písmen. Porovnání provádí pomocí algoritmu založeného na podobnosti histogramů (popsáno níže). Jako výsledek vrátí informaci, o který typ CAPTCHA kódu se jedná.

[3] Příkladem názvu souboru může být myOwnType_ABCD.bmp

[4] Jedná se o počet černých pixelů v každém sloupci písmene

- **Klasifikace založená na podobnosti histogramů.**

  Rozpoznávání se provádí pomocí podobnosti histogramů jednotlivých písmen / tvarů. Z fáze segmentace dostává buď jeden histogram ke každému písmenu nebo všechny varianty ke každému písmenu. To nastane v přípípadě, že se musely oddělit písmena od sebe. Základem pro klasifikaci je algoritmus pro porovnání dvou stejně širokých histogramů [16]. Algoritmus je však upraven, aby byl schopen procentuálně ohodnotit shodnost dvou libovolně širokých histogramů. Základ úpravy algoritmu spočívá v tom, že se každý histogram uměle rozšíří do šířky, aby vznikly dva stejně široké histogramy. A následně se původně užší histogram posouvá pod původně širším a při každém posunutí se spočítá procentuální podobnost pro dva stejně dlouhé histogramy. Po vyzkoušení všech variant se získá nejlepší procentuální podobnost. Tento upravený algoritmus se aplikuje na všechny varianty všech písmen / tvarů. Písmeno / tvar s největší shodou je prohlášeno za znak nacházející se na obrázku.

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \overline{H_1})(H_2(I) - \overline{H_2})}{\sqrt{\sum_I (H_1(I) - \overline{H_1})^2 \sum_I (H_2(I) - \overline{H_2})^2}}. \tag{4.3}$$

  kde

$$\overline{H_k} = \frac{1}{N} \sum_J H_k(J). \tag{4.4}$$

  N je celkový počet prvků histogramu.

Rovnice 4.3 matematicky popisuje porovnání dvou histogramů o stejném počtu prvků. Rovnice 4.4 vyjadřuje průměr všech hodnot v daném histogramu.

4.2 Aplikace pro stahování obrázků z webových stránek

Celá implementace Python skriptu probíhala ve vývojovém prostředí Microsoft Visual Studio 2013. Visual Studio disponuje kvalitním ladícím nástrojem, který je při implementaci velice žádoucí.

Při spuštění skript převezme a zkontroluje vstupní parametry. Z parametrů získá zadanou URL adresu a pokusí se na ní připojit. Pokud je URL adresa zadaná správně a skript se připojí, vrátí informace o připojení a data z požadované stránky. Pomocí dat stažených z webové stránky se získají URL adresy všech obrázků. Před stažením každého obrázku ze seznamu URL se skript podívá na filtr na obrázky, který byl zadán v parametrech. Filtr může být nastaven na typ a velikost obrázku. Při úspěšném stažení obrázku se vytvoří kopie ve formátu BMP. Je to z toho důvodu, že aplikace pro rozpoznávání CAPTCHA kódů pracuje pouze s tímto formátem. Na konci skript vypíše všechny cesty ke staženým souborům, které stahuje do dočasné složky.

Hlavní použité knihovny

- **argparse**

  Knihovna argparse slouží k parsování vstupních parametrů. Umožňuje jednoduchou správu nad vstupními parametry.

- **requests**

  Knihovna pro komunikaci přes protokoly HTTP a HTTPS. Nemusí se explicitně zadávat data do hlaviček protokolu. Stačí pouze vytvořit spojení *requests.Session()* a toto spojení se o vše postará. Jako je již zmíněné vyplnění hlaviček, automatické přesměrování atd. V případě, že komunikace bude probíhat i přes zabezpečený protokol HTTPS, musí se přibalit soubor s certifikáty *cacert.pem*, který ověří důvěryhodnost každého webu pomocí svých certifikátů.

- **re**

  Knihovna umožňující práci s regulárními výrazy. Tyto výrazy se velice vyplatí v získávání informací o URL adresách obrázků ze zdrojového kódu webové stránky.

- **PIL**

  Knihovna poskytující práci s obrazovými daty. Ve skriptu je využita pro formátování stažených obrázků. Konkrétně se jedná o odstraňování alfa kanálu z obrázků, kopírování, či převod libovolného typu obrázku na formát bitmap.

- **cx_Freeze**

  Knihovna poskytující převod Python skriptu (koncovka .py) na spustitelný soubor (koncovka .exe). Tento převod je uskutečněn proto, že cíloví uživatelé nemusí mít ve svém počítači nainstalovaný Python 3.4+ a tím pádem by nešlo využít možnosti stažení obrázků z webů. V případě, že celý skript bude zkompilovaný do spustitelného souboru, nebude uživatel potřebovat mít nainstalované žádné Python knihovny. Všechny potřebné knihovny se přibalí k .exe souboru.

Kapitola 5

Rozšíření

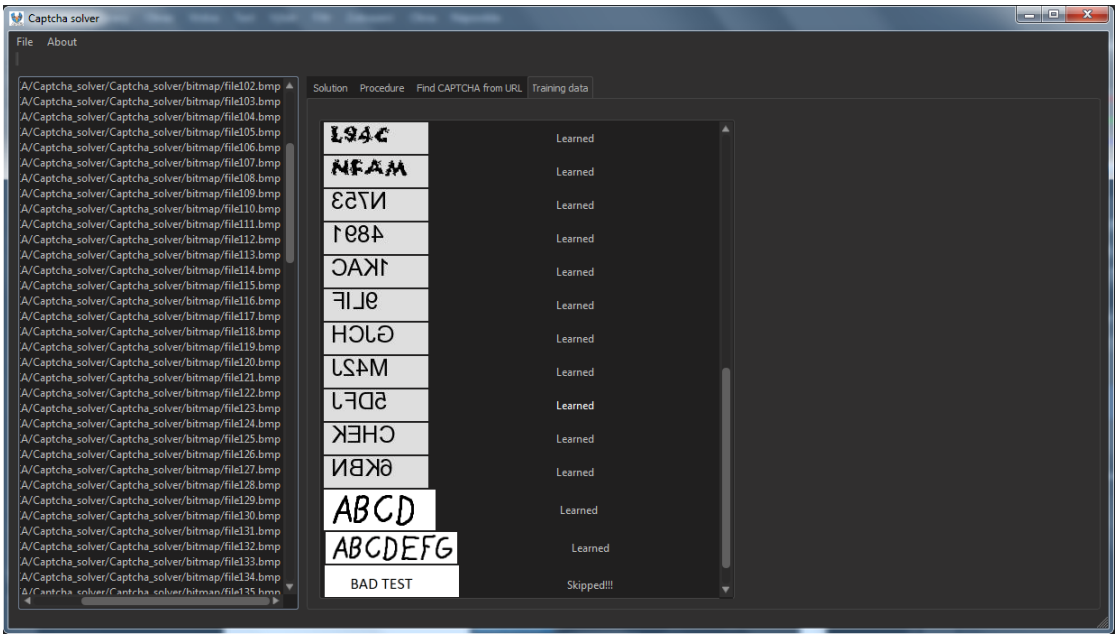
V této kapitole se zaměříme na podrobný popis rozšíření, které bylo vytvořeno nad rámec zadání bakalářské práce.

5.1 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní je navrženo pro zobrazení postupu libovolného CAPTCHA kódu. Např. okna pro zobrazení postupu řešení jsou vytvořena dynamicky, aby bylo možné zobrazit postup rozpoznání CAPTCHA kódu s rozdílným počtem znaků.

5.1.1 Trénování ze sady CAPTCHA kódů

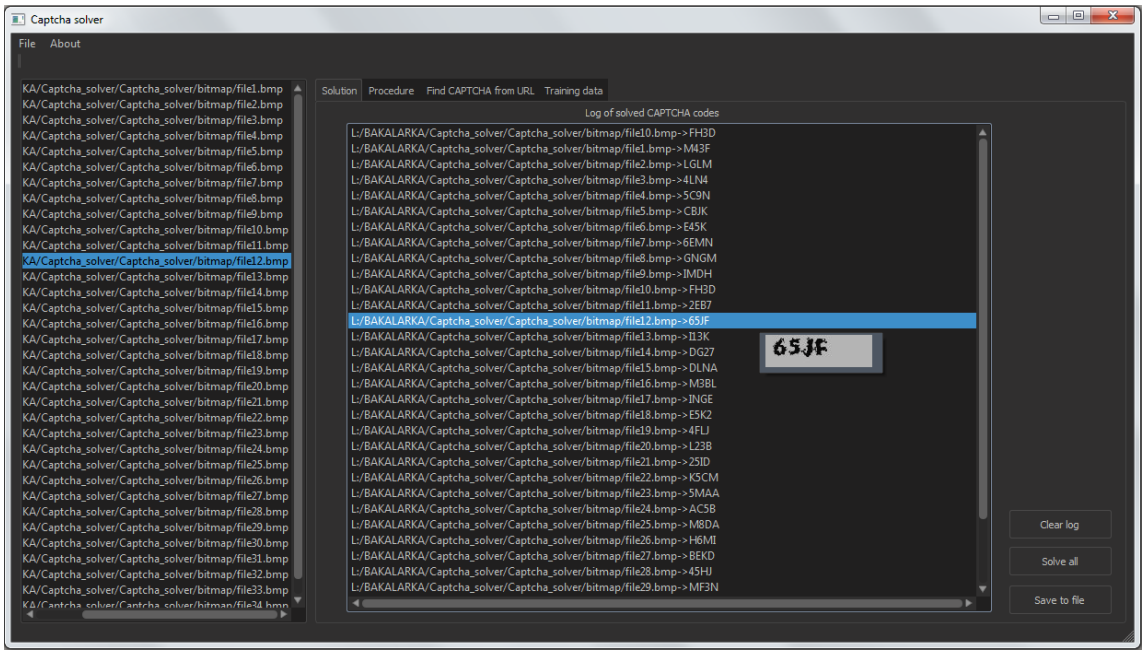
Grafické uživatelské rozhraní umožňuje zobrazit natrénované znaky z CAPTCHA kódů, které se do programu načítají. Zobrazují se v seznamu. U každého CAPTCHA kódu je uvedeno, zda se trénování povedlo nebo zda byl daný CAPTCHA kód z nějakého důvodu přeskočen. Na obrázku 5.1 je zobrazen seznam nově se naučených znaků z CAPTCHA kódů.



Obrázek 5.1: Trénování nových znaků.

5.1.2 Rozpoznávání CAPTCHA kódů

Nejvyužívanější součást grafického uživatelského rozhraní je rozpoznání CAPTCHA kódu bez ukládání postupu. Po načtení složky či souboru s CAPTCHA kódem se každý vybraný soubor přidá do seznamu. Je možnost vyřešit všechny načtené CAPTCHA kódy naráz nebo jednotlivě. Po vyřešení se zaznamenají výsledky do záznamového seznamu. Tento seznam lze uložit do souboru. Jako kontrola správnosti poslouží popisek s obrázkem daného kódu, který se zobrazí po najetí myší na záznam o vyřešení CAPTCHA kódu. Na obrázku 5.2 je zobrazený seznam vyřešených CAPTCHA kódů s popiskem pro kontrolu správnosti.

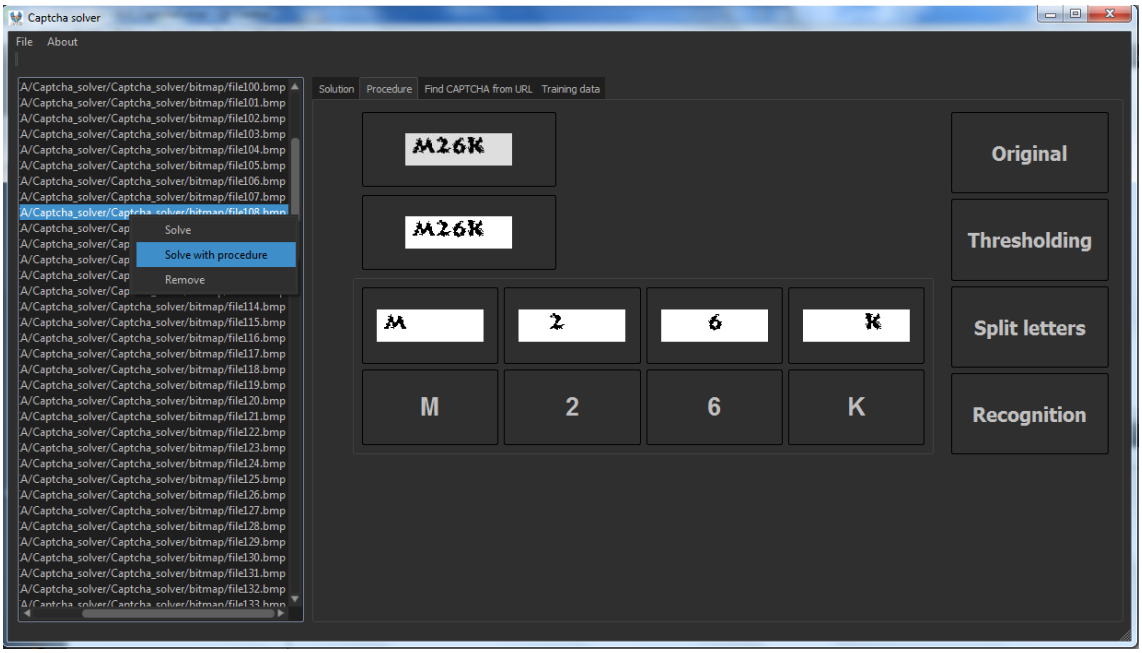


Obrázek 5.2: Řešení CAPTCHA kódů.

5.1.3 Rozpoznávání CAPTCHA kódů s postupem

Další možností grafického uživatelského rozhraní je vyřešení CAPTCHA kódu s postupem. Zobrazuje originální CAPTCHA kód, kód po průchodu fází předzpracování, segmentací a klasifikací.

Při zadání pokynu k vyřešení CAPTCHA kódu s postupem dostane algoritmus cestu k řešenému souboru. Po úspěšném načtení souboru se spočítá jeho md5 hash [21] a zkontroluje se, jestli v dočasné složce neexistuje složka s tímto názvem. Pokud ne, tak se vytvoří a zavolají se funkce pro vyřešení CAPTCHA kódu s informací, že se mají jednotlivé kroky rozpoznávání ukládat do vytvořené složky viz implementace v Kapitole 4. V případě, že již složka se jménem md5 hashe souboru existuje, tak se pouze jednotlivé kroky postupného rozpoznávání zobrazí. Tento algoritmus je zvolen z důvodu optimalizace. Obrázek 5.3 zobrazuje jednotlivé kroky rozpoznávání.

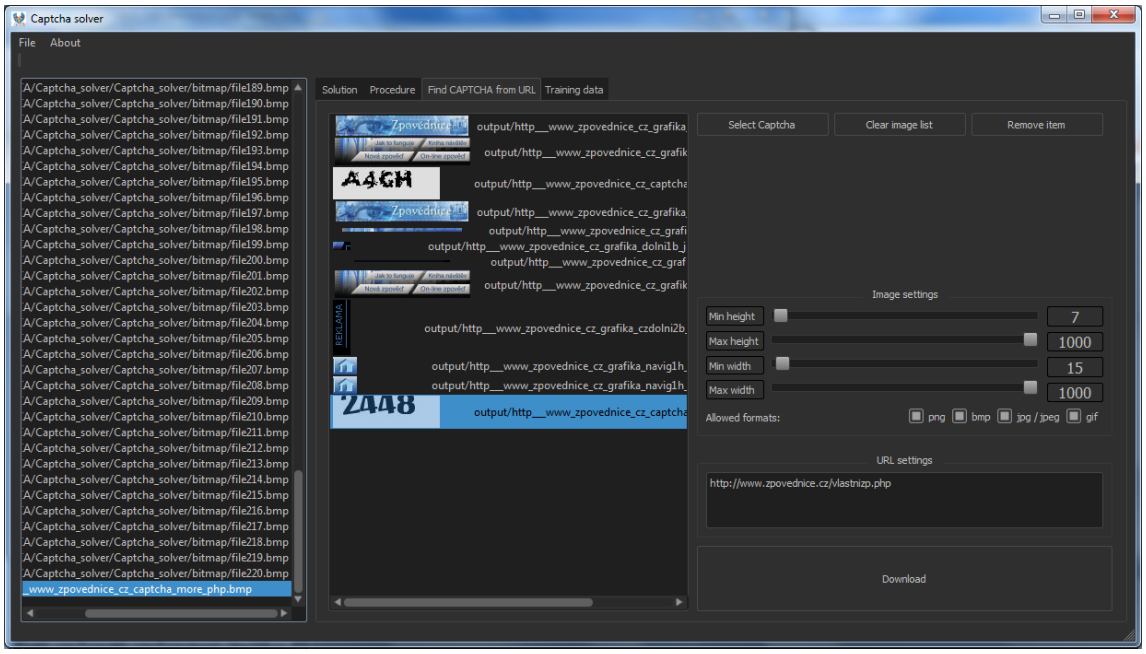


Obrázek 5.3: Řešení CAPTCHA kódů s postupem.

5.1.4 Editor stažených obrázků

Závěrečná funkcionalita grafického uživatelského rozhraní je možnost vyhledání CAPTCHA kódů na webových stránkách a následné přidání do seznamu CAPTCHA kódů určených pro rozpoznání.

Při potvrzení URL tlačítkem „Download“ aplikace zavolá již vytvořený skript a předá mu všechny požadavky, které si uživatel vybral ve formuláři tvořený zaškrtávacími políčky pro výběr formátu a čtyřmi posuvníky pro nastavení povolené velikosti. Po stažení obrázků skript vrátí cesty k souborům a grafické uživatelské rozhraní je zobrazí do seznamu s malými ikonami pro lepší orientaci. Obrázek 5.4 reprezentuje seznam stažených obrázků z webové stránky. Zadaná webová stránka obsahuje i naučený CAPTCHA kód, který můžeme hned rozpoznat.



Obrázek 5.4: Stažení obrázků z webových stránek.

Kapitola 6

Testování

6.1 Testování správnosti rozpoznání kódu

Pomocí testovací sady budeme moci zjistit, na kolik procent je výsledný program schopný řešit daný typ CAPTCHA kódu.

- **Stažení testovací sady**

  Prvním krokem bylo stažení velkého množství CAPTCHA kódů. Na tento úkol posloužil skript, který se přihlásil na webovou stránku obsahující generátor daného CAPTCHA kódu a do vybrané složky stáhl požadovaný počet CAPTCHA kódů. Po stažení originálních kódů je skript převedl do formátu bitmap.

- **Úprava stažené sady**

  Pro testy je nutné dopředu vědět, jaký správný text CHAPTCHA obsahuje. Abychom byli schopni porovnat správné řešení s výsledkem aplikace, je nutné si ho někam zaznamenat. Ideálním řešením bylo přejmenovat každý stažený soubor na text, který CAPTCHA obsahuje.

Vyhodnocení řešeného typu CAPTCHA kódu se provádělo na vzorku 200 náhodně stažených CAPTCHA kódů. Na testované sadě měla aplikace 100% úspěšnost správného rozpoznání.

Aplikace byla také testována na CAPTCHA kódech využívající firma O_2. Na vzorku 200 náhodně stažených CAPTCHA kódech měla aplikace úspěšnost 68 %, což označilo daný CAPTCHA kód za nedostatečný. Na obrázku 6.1 je ukázka O_2 CAPTCHA kódu.

Obrázek 6.1: O_2 CAPTCHA kód.

6.2 Testování GUI na uživatelích

Uživatelé jsou testováni na určitých úkonech s grafickým rozhraním a zjišťujeme jaké úkony jsou pochopitelné a lehko proveditelné a které tvoří problém.

Studentka z farmaceutické fakulty. Na první pohled uživatelka nevěděla, co má dělat, aby se jí zobrazilo správné přečtení kódu. Po přečtení manuálu správně načetla trénovací sadu a následně i testovací CAPTCHA kódy. Chvilku trvalo, než našla tlačítko pro vyřešení načtených CAPTCHA kódů. Po nalezení tlačítka pro vyřešení, bez problémů uložila výsledek do souboru. Při stahování CAPTCHA kódů z webových stránek nebyl žádný problém. Uživatelka si správně zadala formáty požadovaných obrázků, URL a následně stáhla dané obrázky.

Jako výhodu uživatelka uvedla stahování CAPTCHA kódů z libovolných webů. Zaujalo jí také vyřešení CAPTCHA kódu s postupem.

Ing. z humanitního oboru. Načtení trénovací a testovací sady proběhlo bez problémů. Největší problém činilo přinutit program řešit načtené CAPTCHA kódy. Práce s webovými stránkami problém nečinil.

Uživatel 50 let+.

Po pochopení zadání chvilku trvalo, špatně byl uživatel rozeznal funkcionální rozdíl mezi „Open“ a „Open training data“. Dále při načtení trénovací sady a testovací sady nebyl problém. Při řešení testovacího souboru se naskytl problém, že uživatel nemohl najít postup k tomu, aby program začal řešit jednotlivé CAPTCHA kódy. Při práci na stažení obrázků / CAPTCHA kódů z webových stránek nevznikl žádný problém.

Studenti IT oboru.

Po spuštění aplikace se každý student spadající do této kategorie rychle zorientoval. Hned pochopil, co aplikace potřebuje za vstupní data a správně je načetl. Při řešení zadaných CAPTCHA kódů také nevznikl žádný problém.

Seznam testovaných úkonů.

- Načtení trénovací sady.
- Načtení testovacích dat.
- Vyřešení načteného CAPTCHA kódu.
- Vyřešení všech CAPTCHA kódů a následné uložení výsledku do souboru.
- Zobrazení postupu s řešením.
- Nastavení parametrů pro stahování CAPTCHA kódů z URL.
- Stažení CAPTCHA kódů a jiných obrázků z URL.
- Práce se staženými daty.

6.2.1 Náměty na zlepšení aplikace od uživatelů:

- Přejmenování tlačítka „Open“ na „Open test data“ z důvodu lepšího pochopení, co se načítá za data.
- Nápověda, která by pomohla při prvním pochopení programu.
- Převážně uživatelé starší 50 let postrádali český překlad v názvech tlačítek a popisků.
- Více zviditelnit možnost vyřešení načteného CAPTCHA kódu. Uživatelům chvilku trvalo, než našli možnost „Solve“ - nachází se po pravém kliknutí na načtený soubor.

Závěr

V rámci bakalářské práce jsme se seznámili s obecně s CAPTCHA kódem. U textového CAPTCHA kódu jsme se seznámili s jeho bezpečnostními prvky a navrhli jsme, jak se vyvarovat lehko prolomitelným CAPTCHA kódům. Také jsme si uvedli, jaké existují řešení rozpoznání CAPTCHA kódů.

Jako hlavní cíl bakalářské práce byla implementace systému pro řešení CAPTCHA kódů. Za výhodu implementovaného systému se považovala vysoká rychlost učení se nové trénovací sady a následné rozpoznání písmen v daném CAPTCHA kódu v porovnání s existujícím řešením viz Kapitola 1.3. Rychlost učení a rozpoznávání jednoho CAPTCHA kódu se pochybovala v řádech jednotek milisekund. Na pečlivě připravené trénovací sadě, která obsahovala všechna písmena / tvary, neměl systém problém s rozpoznáním. Testování proběhlo na 200 náhodně stažených CATPCHA kódech z fóra Zpovědnice [22]. Úspěšnost rozpoznání se vyšplhala na 100 %. Systém byl také testovaný na CAPTCHA kódech z firmy O_2 [15], u kterého se úspěšnost vyšplhala na 68 %. U obou testovaných CAPTCHA kódů jsme dostatečně překročili hranici 1 %, což znamená, že oba CAPTCHA kódy jsme označili za nevhodné pro použití na webových stránkách.

V případě, že není k dispozici vhodná trénovací sada, rozpoznání CAPTCHA kódu nemusí být vždy správné. Nebo pokud se jedná o CAPTCHA kód se spojenými písmeny, jejichž segmentace je obtížná.

Aplikace by mohla být rozšířena o další možné algoritmy. Vylepšení segmentace - výměna algoritmu semínkového vyplnění za jiné algoritmy využívající konvoluci k detekci hran písmen. Také by se mohl přidat klasifikační algoritmus např. algoritmus K-Nearest Neighbors. Tyto úpravy by nebylo těžké do kódu aplikace přidat z důvodu dodržení dobrého návrhu celé aplikace. Grafické uživatelské rozhraní by mohlo být rozšířeno o editor JSON souboru. Umožňoval by přehledné zobrazení a editaci naučených znaků.

Literatura

[1] 2Captcha: Online CAPTCHA Solving and Image Recognition Service. https://2captcha.com/, 2016, [Online; navštíveno 19-04-2016].

[2] 7 Best CAPTCHA Solvers: 7 Best CAPTCHA Solvers. http://www.slideshare.net/marriagenamechange/7-best-captcha-solvers, 2016, [Online; navštíveno 09-04-2016].

[3] Adobe: https://get.adobe.com/cz/flashplayer/, 2016, [Online; navštíveno 10-05-2016].

[4] Bennamoun, M.; Mamic, G. J.: *Object Recognition: Fundamentals and Case Studies*, kapitola Optical Character Recognition. London: Springer London, 2002, ISBN 978-1-4471-3722-1, s. 199–220, doi:10.1007/978-1-4471-3722-1_5.
URL http://dx.doi.org/10.1007/978-1-4471-3722-1_5

[5] Bursztein, E.; Martin, M.; Mitchell, J.: Text-based CAPTCHA Strengths and Weaknesses. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, New York, NY, USA: ACM, 2011, ISBN 978-1-4503-0948-6, s. 125–138, doi:10.1145/2046707.2046724.
URL http://doi.acm.org/10.1145/2046707.2046724

[6] CAPTCHA: The Official CAPTCHA Site. http://www.captcha.net/, 2016, [Online; navštíveno 17-03-2016].

[7] Chandavale, A. A.; Sapkal, A.: *Recent Trends in Computer Networks and Distributed Systems Security: International Conference, SNDS 2012, Trivandrum, India, October 11-12, 2012. Proceedings*, kapitola Security Analysis of CAPTCHA. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ISBN 978-3-642-34135-9, s. 97–109, doi:10.1007/978-3-642-34135-9_10.
URL http://dx.doi.org/10.1007/978-3-642-34135-9_10

[8] Edge Detection MATLAB: Edge Detection - MATLAB. http://www.mathworks.com/discovery/edge-detection.html, 2016, [Online; navštíveno 06-04-2016].

[9] Harrabi, R.; Ben Braiek, E.: Color image segmentation using multi-level thresholding approach and data fusion techniques: application in the breast cancer cells images. *EURASIP Journal on Image and Video Processing*, ročník 2012, č. 1, 2012: s. 1–11, ISSN 1687-5281, doi:10.1186/1687-5281-2012-11.
URL http://dx.doi.org/10.1186/1687-5281-2012-11

[10] JSON: JSON. http://www.json.org/json-cz.html, 2016, [Online; navštíveno 02-03-2016].

[11] JsonCpp: JsonCpp - JSON data format manipulation library. http://jsoncpp.sourceforge.net/old.html, 2016, [Online; navštíveno 04-04-2016].

[12] Kabai, L.: CAPTCHA. http://wiki.knihovna.cz/index.php/CAPTCHA, 2016, [Online; navštíveno 02-05-2016].

[13] Kramer, O.: *Dimensionality Reduction with Unsupervised Nearest Neighbors*, kapitola K-Nearest Neighbors. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ISBN 978-3-642-38652-7, s. 13–23, doi:10.1007/978-3-642-38652-7_2.
URL http://dx.doi.org/10.1007/978-3-642-38652-7_2

[14] Lee, J.; Kang, H.: Flood fill mean shift: A robust segmentation algorithm. *International Journal of Control, Automation and Systems*, ročník 8, č. 6, 2010: s. 1313–1319, ISSN 2005-4092, doi:10.1007/s12555-010-0617-6.
URL http://dx.doi.org/10.1007/s12555-010-0617-6

[15] o2: https://moje.o2.cz/firma/simpleImg, 2016, [Online; navštíveno 10-04-2016].

[16] OpenCV: Histograms OpenCV 2.4.12.0 documentation. http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html, 2016, [Online; navštíveno 28-03-2016].

[17] Rolko, J.: 3D captcha. http://3dcaptcha.net/, 2016, [Online; navštíveno 01-05-2016].

[18] Saravanan, C.: Color Image to Grayscale Image Conversion. In *Computer Engineering and Applications (ICCEA), 2010 Second International Conference on*, ročník 2, March 2010, s. 196–199, doi:10.1109/ICCEA.2010.192.

[19] sladkymeda: http://www.sladkymeda.cz/captcha, 2016, [Online; navštíveno 10-04-2016].

[20] Suphannee Sivakorn Jason Polakis Angelos D Keromytis: I'm not a human: Breaking the Google reCAPTCHA. https://www.blackhat.com/docs/asia-16/materials/asia-16-Sivakorn-Im-Not-a-Human-Breaking-the-Google-reCAPTCHA-wp.pdf, 2016, [Online; navštíveno 06-05-2016].

[21] van Tilborg, H. C. A.; Jajodia, S. (editoři): *Encyclopedia of Cryptography and Security*, kapitola MD5 Hash Function. Boston, MA: Springer US, 2011, ISBN 978-1-4419-5906-5, s. 771–771, doi:10.1007/978-1-4419-5906-5_1197.
URL http://dx.doi.org/10.1007/978-1-4419-5906-5_1197

[22] Zpovědnice: http://www.zpovednice.cz/captcha_more.php, 2016, [Online; navštíveno 10-04-2016].

[23] Černín, Z.: Captcha - jak znechutit roboty a neodradit uživatele. http://www.cognito.cz/technologie/captcha-jak-znechutit-roboty-a-neodradit-uzivatele/, 2016, [Online; navštíveno 03-04-2016].

Příloha A

Obsah CD

- **bin_win32** - složka obsahující spustitelnou aplikaci, včetně všech potřebných knihoven
- **data** - složka obsahuje trénovací a testovací sadu CAPTCHA kódů
- **src** - složka obsahuje všechny zdrojové soubory
- **src_latex** - složka obsahující zdrojové kódy písemné práce
- **poster.pdf** - plakát prezentující práci, její cíle a dosažené výsledky
- **manual** - Uživatelská příručka: Spuštění a práce s aplikací

Příloha B

Manual

B.1 Příprava aplikace ke spuštění

V této kapitole si uvedeme, co je nutné udělat, aby jsme byli schopni přeložit zdrojové kódy aplikace do spustitelné podoby. V první řadě si popíšeme přeložení grafického uživatelského rozhraní a následně si připravíme skript pro stahování obrázků z webových stránek.

B.1.1 Grafické uživatelské rozhraní

Ke kompilaci grafického uživatelského rozhraní je nutné mít nainstalovaný program Qt Creator, který dokáže přeložit celý projekt. Aplikace byla vytvořena v Qt Creatoru verze Qt 5.6.0 (MSVC 2013, 32 bit). Po kompilaci se vytvoří složka „release“ do které je nutno nakopírovat dynamické knihovny (dll), které aplikace vyžaduje. Tyto knihovny se nachází ve složce s nainstalovaným Qt. Aplikace vyžaduje tyto knihovny (je možné si potřebné knihovny zkopírovat ze složky bin_win32):

- libEGL.dll
- libgcc_s_dw2-1.dll
- libstdc++-6.dll
- libwinpthread-1.dll
- Qt5Core.dll
- Qt5Gui.dll
- Qt5Widgets.dll
- qwindows.dll - tato knihovna se musí umístit to složky „platforms“

Po přípravě všech knihoven se musí vytvořit dvě složky „output“ pro obrázky stažené z webových stánek a „tmp“ pro dočasné soubory. Také se musí vytvořit soubor „histogramsData.json“ a do něj zapsat „{}“

B.1.2 Skript ke stahování obrázků

Zkompilovaná aplikace z Kapitoly B.1.1 očekává ve své složce i zkompilovaný python skript do spustitelné podoby (exe soubor).

Kompilace Python skriptu

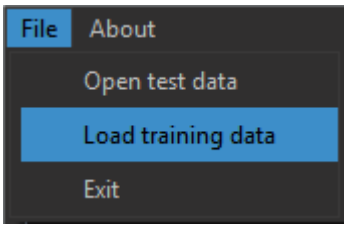
Je nutné mít nainstalovaný Python 3.4 a vyšší.

1. Do složky „Python34“ nakopírujeme obsah složky „Images downloader“.
2. Spustíme příkazový řádek (cmd) v této složce.
3. Spustíme příkaz „python.exe setup.py build“
4. Ve složce „./build/exe.win32-3.4“ nalezneme zkompilovaný skript do spustitelné podoby.
5. Obsah složky nakopíruje do složky „release“, ve které se nachází připravená aplikace z Kapitoly B.1.1.

B.2 Práce s aplikací

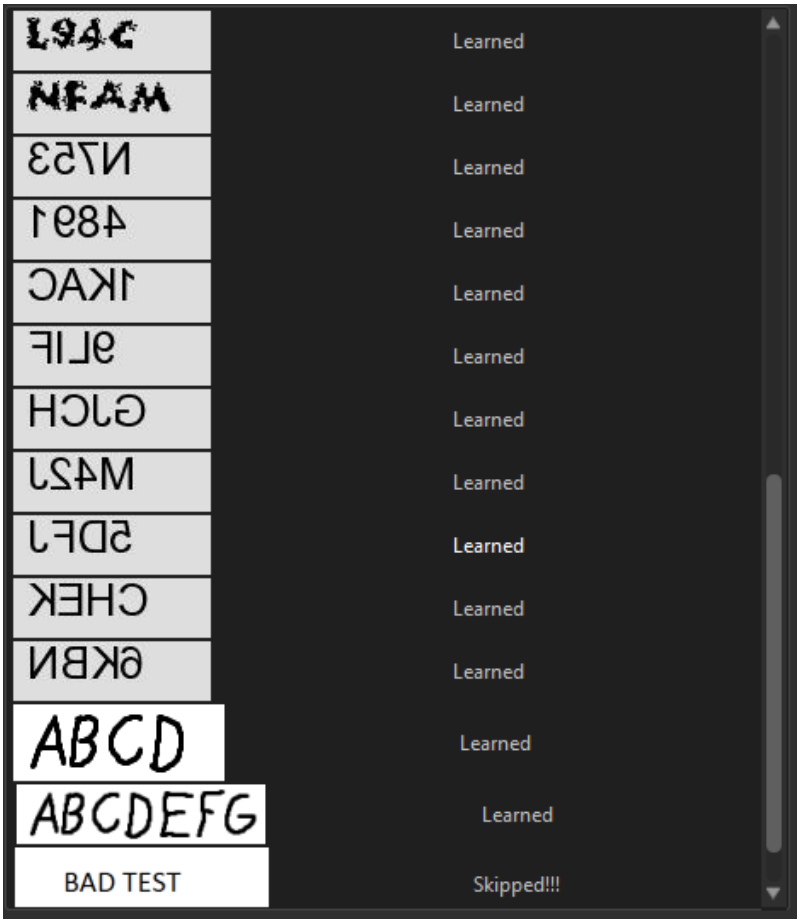
Tento manuál slouží k prvnímu seznámení s aplikací. Je představena základní funkcionalita.

Prvním krokem pro práci s programem je načíst trénovací sadu CAPTCHA kódů viz B.1. Bez trénovací sady by měl program veliké problémy přečíst CAPTCHA kódy.



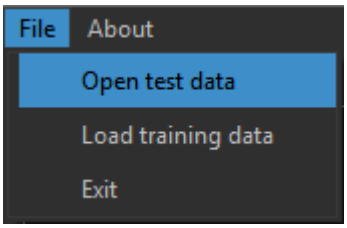
Obrázek B.1: Otevřít trénovací sadu.

Po načtení trénovací sady se zobrazí seznam naučených znaků viz B.2.



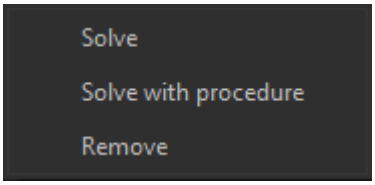
Obrázek B.2: Seznam naučených znaků.

Po natrénování sady CAPTCHA kódů je program schopný rozpoznávat daný typ. Můžeme si načíst testovací data.



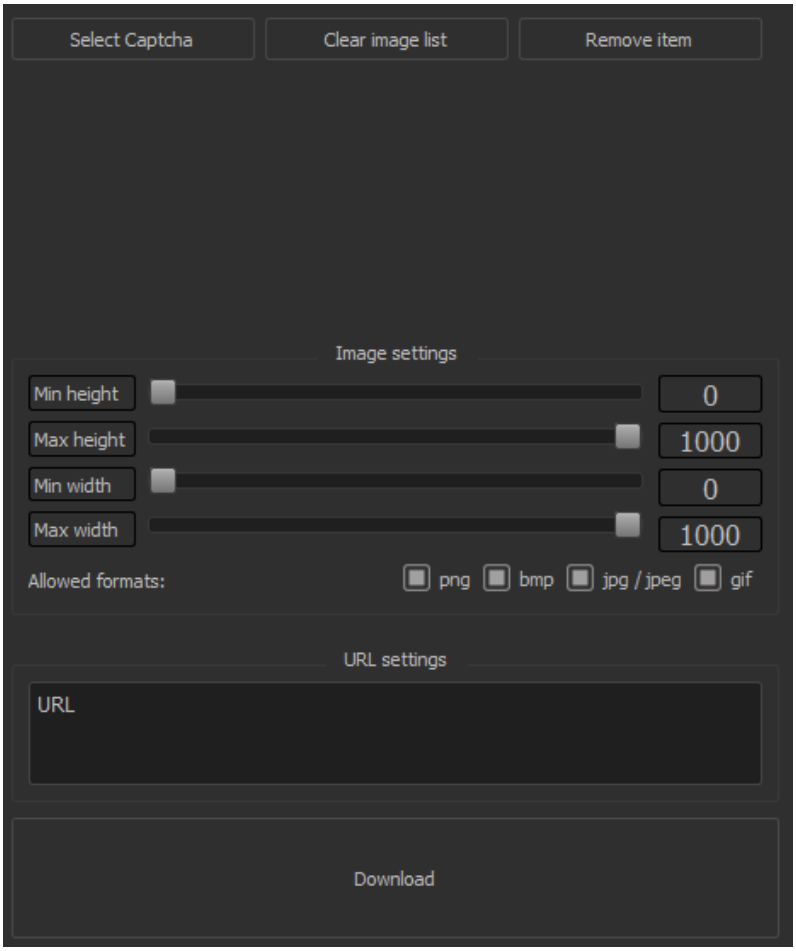
Obrázek B.3: Načtení testovacích CAPTCHA kódů.

Načtené CAPTCHA kódy se zobrazí v levém seznamu. Při kliknutí pravým tlačítkem na načtený CAPTCHA kód můžeme rozpoznat daný CAPTCHA kód rozpoznat bez nebo s postupem. Na obrázku B.4 je zobrazeno kontextové menu, pro výběr řešení.



Obrázek B.4: Kontextové menu řešení CAPTCHA kódů.

Aplikace také umožňuje stažení CAPTCHA kódů z webových stránek. Na obrázku B.5 je uveden formulář pro nastavení parametrů pro stahované obrázky z webových zdrojů. Může se nastavit povolená maximální a minimální šířka a výška obrázku a formát.



Obrázek B.5: Nastavení parametrů pro obrázek.

Po zadání URL a stisku tlačítka „Download“ se začnou obrázky stahovat. Po chvíli se zobrazí v seznamu uprostřed programu. Výběr CAPTCHA kódu se provádí označením požadovaného obrázku v seznamu a stiskem tlačítka „Select Captcha“. Požadovaný obrázek se zobrazí v seznamu načtených CAPTCHA kódů. Poté je možné daný CAPTCHA kód rozpoznat.