VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ZPRACOVÁNÍ OBRAZU A AUTOMATICKÉ ŘEŠENÍ NONOGRAMŮ

IMAGE PROCESSING AND AUTOMATIC SOLVING OF NONOGRAMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Bálint Udvardy
AUTHOR

VEDOUCÍ PRÁCE Ing. Ilona Janáková, Ph.D.
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Bálint Udvardy ***ID:*** 186217
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Zpracování obrazu a automatické řešení nonogramů

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je automaticky řešit logické hlavolamy, tzv. nonogramy (malované křížovky, kódované obrázky), kdy vstupem vytvořeného algoritmu je fotografie vytištěného hlavolamu.

1. Proveďte rešerši dané problematiky.
2. Navrhněte robustní algoritmy detekce mřížky a čísel hlavolamu z fotografie a řešte automatické rozpoznání čísel. Výsledky algoritmů ověřte na dostatečně široké a pestré databázi snímků.
3. Seznamte se s existujícími postupy automatického řešení nonogramů, případně navrhněte vlastní. Vybranou metodu implementujte.
4. Doplňte úlohu i o variantu detekce částečně a úplně ručně vyřešených hlavolamů.
5. Vytvořte uživatelskou aplikaci, která bude vedle pořízení snímku a generování výsledného řešení nonogramu umožňovat také různé režimy, např. režim nápovědy či ověření a oprav částečně vyluštěné křížovky.
6. Zhodnoťte výsledky.

DOPORUČENÁ LITERATURA:

HLAVÁČ, V., ŠONKA, M.: Počítačové vidění. Praha: Grada, 1992, ISBN 80-85424-67-3.

SALCEDO-SANZ, Sancho, et al. Teaching advanced features of evolutionary algorithms using Japanese puzzles. IEEE Transactions on Education, 2007, 50.2: 151-156.

Termín zadání: 5. 2. 2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: Ing. Ilona Janáková, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta elektrotechniky a komunikačních technologií Ústav automatizace a měřicí techniky

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Abstrakt

V některých novinách se často vyskytují hlavolamy různého typu. Nejčastější se jedná švédské křížovky a sudoku, ale také se setkáme někdy s malovanými křížovkami. Malované křížovky vyžadují pro řešení matematickou logiku, což je výhodné pro počítačové algoritmy. Cílem této práce je vytvořit algoritmus, který vyřeší tyto křížovky z digitálního snímku vytvořeného uživatelem pomocí skeneru nebo smartphonu.

Klíčová slova

Malované křížovky, nonogramy, zpracování obrazu, segmentace, nonogram solver

Abstract

In many puzzle books we come across logical puzzles like sudoku or nonograms (also known as griddlers or piccrosses), which demand mathematical logic to solve them. This work focuses on solving nonograms based on digital images of the puzzle taken with a smarthphone or a scanner. This solution is restricted to black and white nonograms, which can be slightly distorted by the angle of the camera during capture.

Keywords

Nonograms, picrosses, image processing, segmentation, puzzle solver, nonogram solver

Bibliografická citace:

UDVARDY, B. Zpracování obrazu a automatické řešení nonogramů . Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. 55s. Vedoucí bakalářské práce Ing. Ilona Janáková, Ph.D

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Zpracování obrazu a automatické řešení nonogramů jsem vypracoval samostatně pod vedením vedoucí bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **21. května 2018** ……
podpis autora

Poděkování

Děkuji vedoucí bakalářské práce Ing. Iloně Janákovej, PhD. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalařské práce.

V Brně dne: **21. května 2018** ……

podpis autora

Obsah

Seznam symbolů a zkratek

Zkratky:

GUI	…	graphical user interface
angl.	…	anglicky
OCR	…	optical character recognition
NaN	…	not a number
č.	…	číslo
DFS	…	depth-first search
CD	…	compact disc

Seznam obrázků

ÚVOD

Tato práce se zabývá vývojem uživatelské aplikace v programu Matlab, která bude schopna vyřešit černobílé malované křížovky pomocí digitální fotografie zadání křížovky. Aplikace bude ve formě grafického uživatelského prostředí (angl. GUI).

Protože malované křížovky nejsou v dnešní době příliš rozšířené, další kapitola obsahuje stručný popis hlavolamů tohoto typu.

Následující kapitoly obsahují podrobnější rozbor problému zpracování obrazu, poněvadž největší část této práce pojednává o získávání dat z pořízeného obrazu.

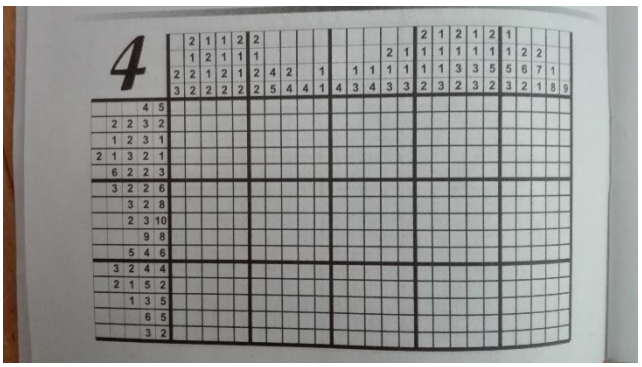
Jestliže všechny informace, které potřebujeme pro numerické řešení problematiky, jsou známy, dojde k číselnému řešení hlavolamu. Tato část se zabývá vytvořením algoritmu, který je pro tento typ hlavolamů a jeho automatické řešení vhodný.

Poslední část práce je zaměřena na grafické řešení hlavolamů pomocí uživatelské aplikace.

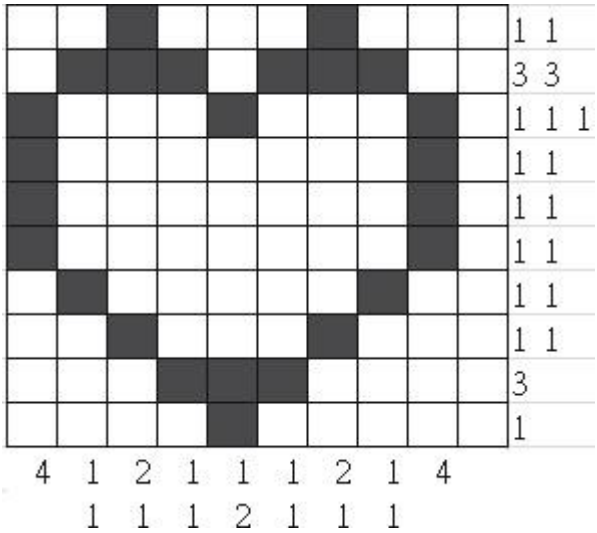
1. ÚVOD O NONOGRAMECH A METODY ŘEŠENÍ

V roce 1987 japonský grafik, Non Ishida vyhrál soutěž v Tokiu pomocí světelného představení, kdy v určitých částech mrakodrapu rozsvěcoval a zhasínal světla. Následně japonský „puzzle maker“ Tetsuya Nishio vyvinul typ hlavolamu podle tohoto vzoru. V Česku se malované křížovky objevují od půlky 90. let v časopise Panorama křížovek.[1]

Malovaná křížovka (někdy také Zakódované obrázky) je logický hlavolam, při kterém je okolo mřížky umístěná legenda s čísly, pomocí které lze získat obrázek. Ve většině případů je výsledek nonogramů černobílý obrázek. Někdy se může jednat i o barevný obrázek, ale tento případ by byl mnohem náročnější vzhledem ke zpracování obrazu a jiným pravidlům, která se týkají řešení. Z tohoto důvodu se tato práce zabývá pouze černobílými nonogramy. Legenda křížovky se nachází většinou v horní a levé části mřížkové oblasti, ale nemusí to být pravidlem (může být i na pravé straně a v dolní části hlavolamu) a číslice nemusí ani být v mřížce.



Obrázek 1: Obrázek nonogramu z měsíčníku[26]



Obrázek 2: Obrázek nonogramu, kde je legenda umístěna na jiném místě než na Obrázku 1

Můžeme říci, že neexistuje žádné definované pravidlo, které by určilo standardní tvar nonogramů, ale pro řešení existují všeobecná pravidla:

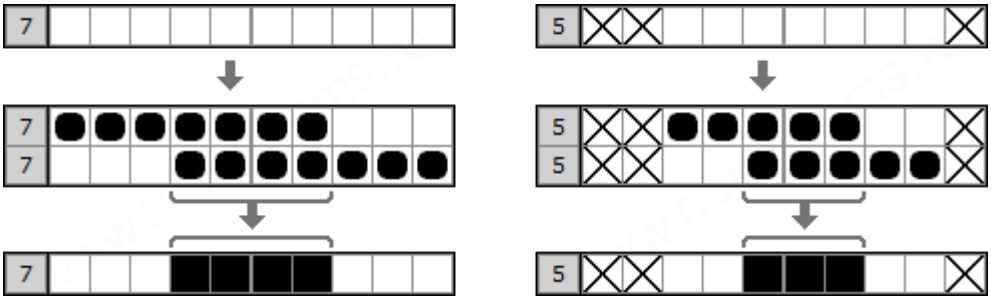
- Každé číslo v legendě určuje počet za sebou následujících čtverečků stejné barvy.
- Mezi jednotlivými čísly je vždy minimálně jeden prázdný čtvereček.
- Čísla v legendě jsou již uspořádána ve správném pořadí.

Pro ruční řešení existují různé metody, které jsou někdy použity i v řešících algoritmech. Tyto metody naleznete v následující kapitole.

1.1 Metody pro řešení [2]

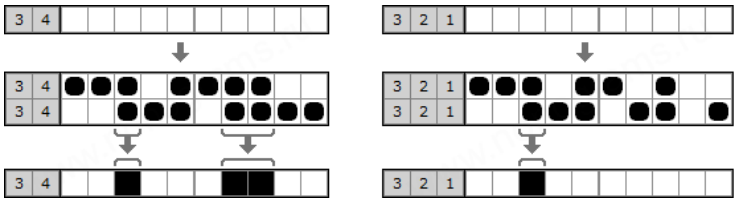
1.1.1 Superpozice extrémních pozic

Pokud řádek nebo sloupec obsahuje jenom jedno číslo, která značí větší délku než je polovina rozměru vyplnitelného řádku nebo sloupce, je jisté, že uprostřed tohoto sloupce či řádku budou některé čtverce vyplněné. Abychom zjistili, konkrétně které části máme vyplnit, je třeba uvažovat extrémní levou a extrémní pravou pozici vyplnitelných čtverců. Místo, kde se prolínají tyto skupiny čtverců, bude ve výsledku vyplněno, jak je ukázáno na Obrázku 3.



Obrázek 3: Ukázka vyplnění pozic podle metody superpozice extrémních pozic

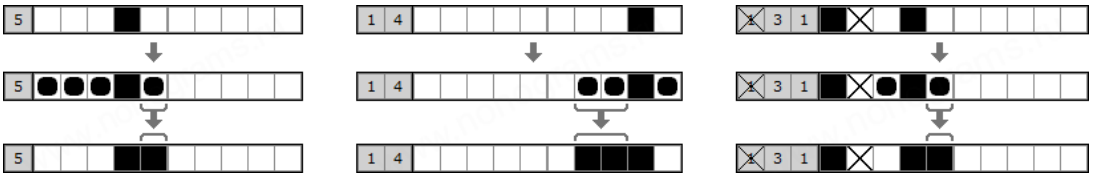
Jestliže řádek nebo sloupec obsahuje několik čísel, můžeme také superponovat krajní levou pozici skupiny čtverců nad krajní pravou, ale můžeme namalovat čtverce pouze na těch místech, kde skupina čtverečků překryje sebe sama (viz Obrázek 4). Měli bychom také vzít v úvahu minimální prostor mezi skupinami čtverců (u černobílých nonogramů je mezi vyplněními skupinami čtverců vždy minimálně jeden prázdný čtvereček).



Obrázek 4: Ukázka metody superpozice extrémních pozic na řádku, které obsahují více čísel než jedno a jejichž součet je větší než poloviční délka řádku

1.1.2 Vyplnění z kraje

Je-li v řadě vybarvený čtverec a vzdálenost, o kterou je levý okraj křížovky kratší, než je potřebná délka pro umístnění bloku čtverců, pak je možné vyplnit několik buněk nalevo od tohoto čtverce. Stejná metoda funguje na poslední číslici a na pravém okraji křížovky.

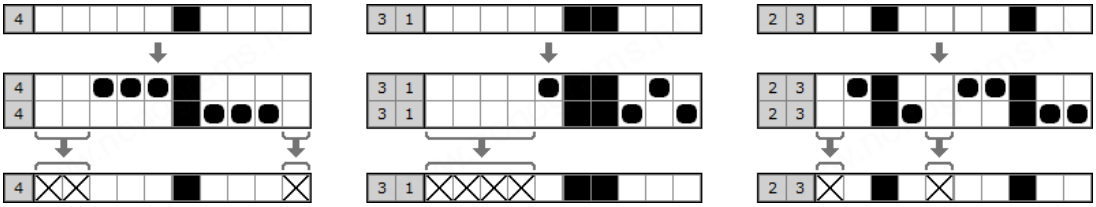


Obrázek 5: Ukázka metody „vyplnění z kraje“

1.1.3 Nepřístupnost

Jestliže v řádku jsou vybarvené čtverce, které jsou určitě odkazovány na konkrétní čísla, je možné umístit kříže na čtvercích „nepřístupných“, které jsou dále než

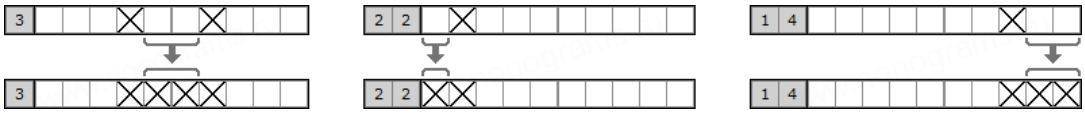
maximální délka, kterou určuje číslo v legendě. Nejčastěji se tato metoda používá u čtverce (nebo několika čtverců), které se mohou týkat pouze prvního nebo posledního čísla.



Obrázek 6: Ukázka metody nepřístupnosti

1.1.4 Metoda „Does not fit“

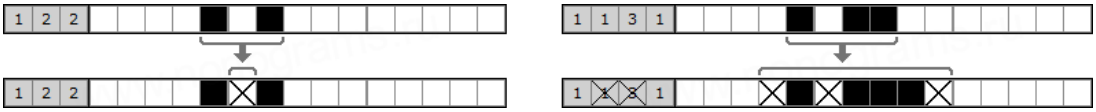
Existují situace, kdy se na řádku objevují oblasti označené křížky, mezi které se nevejde žádná skupina čtverců z té řady. Takové části potom můžeme vyplnit křížky.



Obrázek 7: Ukázka metody „nevejde“

1.1.5 Oddělení

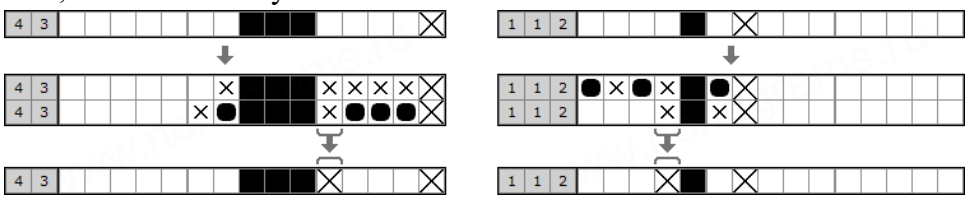
V situacích, kdy jsou některé vyplněné buňky rozděleny prázdnou buňkou, je třeba zkontrolovat, zda může být vyplněna nebo ne. V případě, že to odporuje legendě, buňka musí mít vyplněna křížkem.

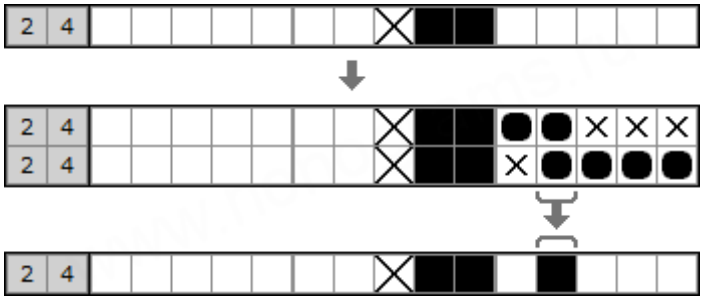


Obrázek 8: Ukázka metody oddělení

1.1.6 Dvojité umístění

Někdy nastane, že namalovaný čtverec v řádku může odkazovat na dvě varianty polohování skupin čtverců. Čtverce, které zůstávají v obou variantách polohování prázdné, označíme křížky.





Obrázek 9: Ukázka metody „dvojité umístění“

2. ZPRACOVÁNÍ OBRAZU A ZÍSKÁVÁNÍ DAT

Jak to bylo řečeno v předchozí kapitole, nonogramy nemají žádné standardní tvary, co se týká rozměrů, umístění legendy, tvaru nebo barvy vyšrafování buňky. Vytvoření obecné algoritmu by bylo velmi obtížné, proto jsou určeny podmínky, které hlavolam musí splnit, aby jej program byl schopný vyřešit.

Předpoklady:

- Nonogram je černobílý.
- Buňky musíme vyplnit celé (ne jenom části buňky).
- Obrázky byly vytvořeny tak, aby obsahovaly celý hlavolam s mírným geometrickým zkreslením (největší část obrazu tvoří samotný hlavolam, který není příliš natočený nebo vyfotografován pod extrémním úhlem zkreslení).

Pro zpracování obrazu je použit program Matlab 2018a.

2.1 Předzpracování obrazu

Cílem předzpracování obrazu je potlačit šum, vady obrazu, odstranit zkreslení, potlačit nebo zvýraznit rysy obrazu a také snížit velikost dat pro další zpracování.[3]

2.1.1 Binarizace obrazu

Cílem této části je snížit množství dat pro další práci s obrazem (zvláště barevné složky a nadbytečné jasové úrovně).

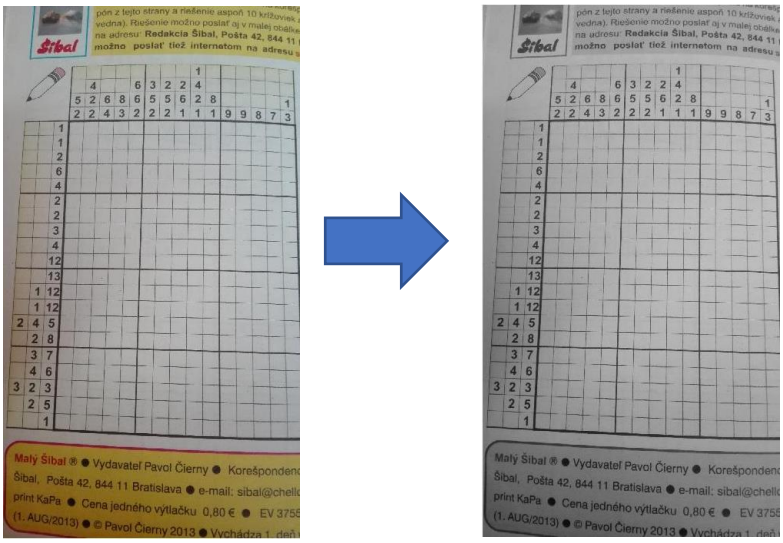
Protože ve většině případů se v dnešní době používají digitální kamery, které generují digitální snímky různých rozměrů, předpokládá se, že vstupní data budou ve standartních formátech, které podporuje použitá verze programu matlab.

Vstupní obraz je tedy obecný barevný obraz. Nejčastěji se používá třísložkový aditivní model RGB a to znamená, že výsledná barva obrazového bodu bude reprezentována pomocí tří barevných složek (červené, zelené a modré barvy). Bitová hloubka nám dává informaci o tom, kolik paměti je potřeba pro ukládání jednoho pixelu jedné barevné složky. Jedna z nejčastěji používané bitové hloubky je osmibitová, která je taky podporována programem Matlab.

2.1.1.1 Odstranění barevných složek

V prvním kroku je třeba odstranit nadbytečné barevné složky obrazu, poněvadž tato práce řeší pouze černobílé hlavolamy. Na to existuje funkce v matlabu „rgb2gray“, která redukuje barevné složky RGB snímku do jedné (odstíny šedé). Funkce používá různé váhy pro transformaci barevných složek, a to v poměru 0,2989/0,5870/0,1140 pro barvy červené, modré a zelené.[4]

2.1.1.2 Prahování



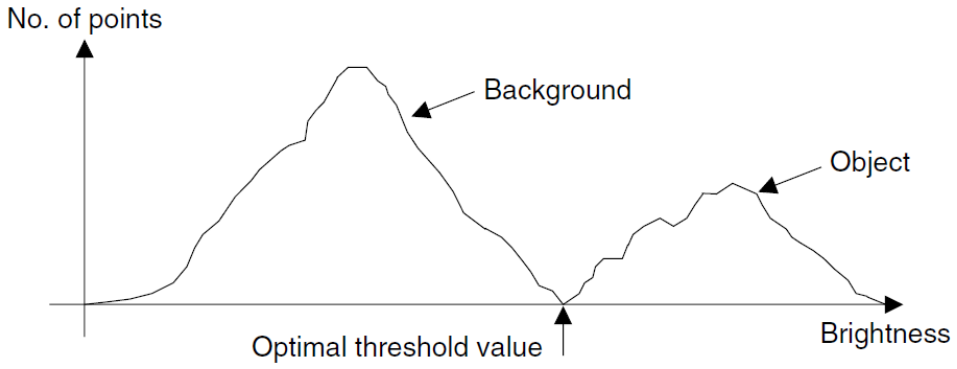
Obrázek 10: Původní obrázek o nonogramu a šedotónový obraz po použití funkce Matlabu „rgb2gray“

Prahování je jasová transformace, která snižuje počet jasových úrovní na nižší (v případě osmibitové jasové hloubky 256) počet, než byl původní. To probíhá podle takzvaných prahů, které byly stanoveny tak, aby došlo ke snížení datových informací, aniž bychom ztratili informace, které považujeme za důležité. Prahy jsou stanoveny z jasových hodnot pixelů původního obrazu, velkou roli hraje histogram obrázku. Rozlišují se dva typy prahování: uniformní (zde se používá jeden práh pro celý obraz) a adaptivní (zde se práh, podle kterého prahujeme, může měnit pro různé pixely).[3]

Protože hlavolam má mřížkovou podobu a ta ve většině případů obsahuje bílé a černé čáry, jsme schopni pomocí prahování na jeden práh obraz snadně binarizovat.

Uniformní prahování

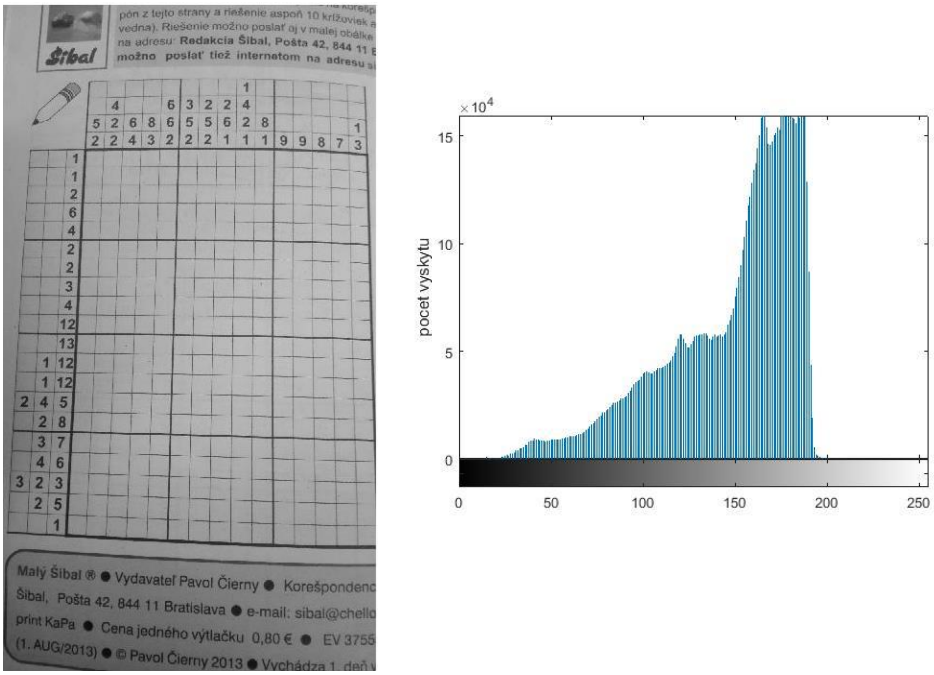
Pro tento typ prahování se používá četnost jasové úrovně pixelů ve snímku, podle které je možné dopočítat vhodný práh.



Obrázek 11: Ukázka zvolení optimální jasové hodnoty pro prahování [5]

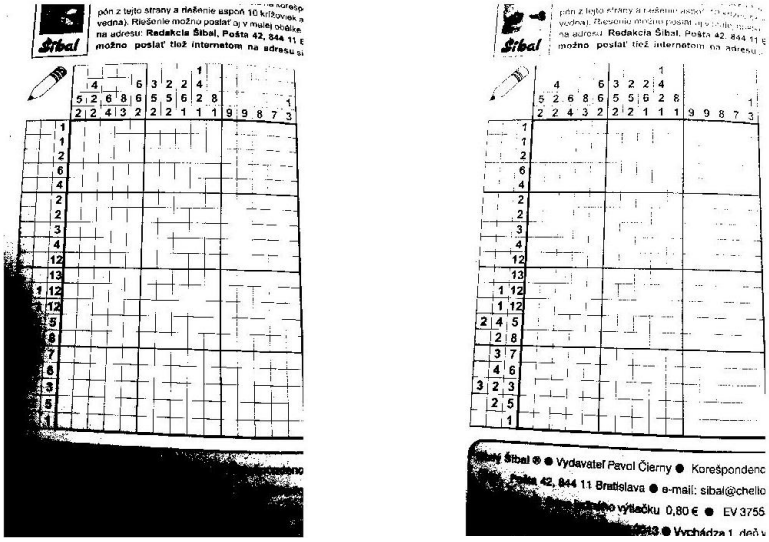
Nejčastěji použitá metoda pro uniformní prahování je Otsuova metoda (Otsu, 1979).

Mějme následující šedotónový obraz křížovky:



Obrázek 12: Šedotónový obraz křížovky (vlevo) a histogram obrazu (vpravo)

Výsledky prahování podle Otsuovy metody a podle empiricky zjištěného prahu jsou uvedeny níže.

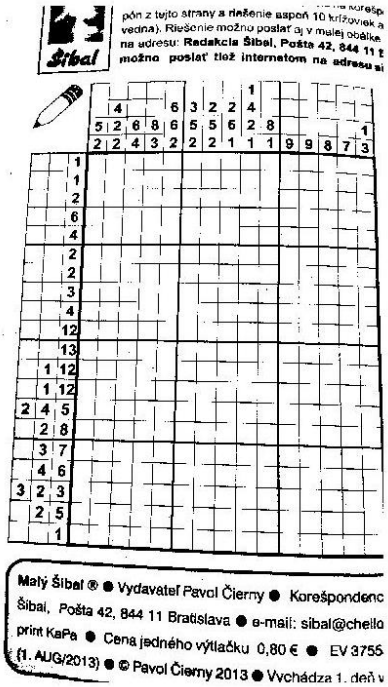


Obrázek 13: Obrázek nonogramu po binarizaci podle Otsuovy metody (vlevo) a podle empiricky stanoveného úrovně (vpravo)

Z obrazků je patrné, že sice pravý snímek vypadá lépe, ale každopádně došlo ke ztrátě důležitých informací, a to z důvodu nerovnoměrného osvětlení objektu. Je zřejmé, že jeden obecný práh, není dostačující pro binarizaci.

Adaptivní prahování

Jak to bylo již řečeno, při adaptivním prahování jasové úrovně, podle které prahujeme, se mohou lišit v závislosti na místě pixelu v obraze a na jasové úrovni v okolí obrazového bodu. V Matlabu existuje funkce pro binarizaci snímků s možností vybrat adaptivní prahování „imbinarize (´adaptive´, …)“ [6], která pracuje na základě Bradleyho metody (D. Bradley and G. Roth, „Adaptive Thresholding Using Integral Image“).[7] Tato metoda používá proměřovací filtr s dostatečně velkým okolím zkoumaného bodu. Když je hodnota vstupního pixelu o několik procent větší než intenzita, kterou udává filtr, tak nastaví nulu. Funkce pracuje s integrálním obrazem, tzn. každý bod ve výsledném obrazu bude mít hodnotu součtu hodnot předchozích pixelů. Vypočítá se adaptivní práh a jako návratovou hodnotu dává již binarizovaný obraz.



Obrázek 14: Snímek po adaptivní binarizaci v matlabu

Jak je vidět, výslední obraz je mnohem jasnější, než předchozí (Obrázek 13). Pro binarizaci snímků je tedy použito adaptivní prahování, neboť funguje mnohem lépe a efektivněji, když není osvětlení objektu rovnoměrné, což je velmi časté u fotografií pořízených smartphonem.

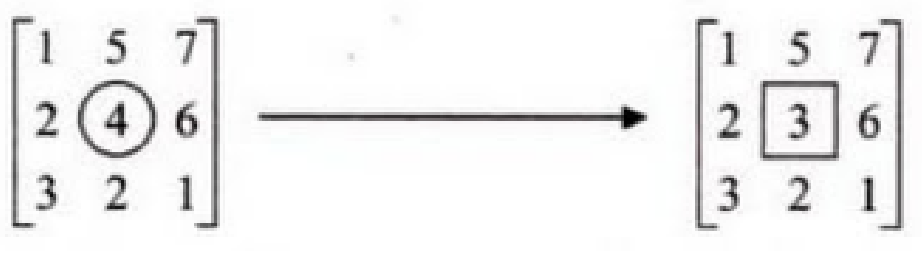
2.1.2 Odstranění šumu a korekce zkreslení

Tato část se zabývá odstraněním vzniklých šumů v binárním obraze a případně i korigovat zkreslení obrazu, které by mohlo ztížit proces segmentace.

2.1.2.1 Odstranění šumu

Šum je nechtěný signál, který nese z hlediska zpracování obrazu bezvýznamnou informaci. Může vzniknout při digitalizaci a přenosu obrazu nebo vlivem vnější magnetického pole. Existují různé typy šumu. Nejčastější typem je Gaussovské, Poissonův a šum typu „pepř a sůl“. Poslední jmenovaný je z hlediska binárního obrazu nejpodstatnější, poněvadž po prahování mohou vnikat místa v obraze, kde se uvnitř černej oblasti objeví náhodný bílý pixel nebo naopak. Pro filtraci šumu v obraze se používají různé filtry. Nejčastěji používané filtry pro tento účel jsou mediánové a morfologické filtry.[8]

Mediánová filtrace
Mediánová filtrace je nelineární filtrace obrazu, kde je jasová hodnota pixelu vypočítaná z jasových hodnot okolních pixelů tak, že tyto hodnoty jsou uspořádané ve vzestupném (nebo v sestupném) pořadí a výsledná hodnota zkoumaného pixelu je medián z těchto hodnot.



Obrázek 15: Původní obraz (vlevo) a výsledek po mediánové filtraci (vpravo)[3]

Tento filtr může výrazně potlačit šum typu „pepř a sůl“, nicméně při zvětšování filtru dojde k odstranění tenkých čar a k rozmazání rohů v obrazu.

Morfologické otevření
Potlačení šumu pomocí morfologické operace nejčastěji používá operace morfologického otevření. Je to operátor získaný kombinací dilatace a eroze; eroze následovaná dilatací se stejným strukturním elementem. Tato operace odděluje objekty spojené tenkou čarou a odstraňuje šum. Druhý jmenovaný je pro případ nonogramů podstatný. Míra rozpojení a redukce šumu je dána velikostí a tvarem strukturního elementu.

Mediánová filtrace (i když používá sort metodu) je rychlejší než morfologické otevření, které musí pro každý pixel vypočítat erozi i dilataci. Z tohoto důvodu je použita funkce Matlabu „meadfilt2“. Tato funkce bez udávání dalších parametru provede mediánovou filtraci daného obrazu čtvercovou maskou 3x3.[9] Při testování na databázi obrazů se ukázalo, že je toto řešení dostatečné. U čar, které jsou alespoň 2 pixely široké (tato podmínka je většinou splněna), nedojde k jejich přerušení, a protože je filtrován binární obraz i uspořádání pole jedniček a nul, nejedná se o složitý proces.

2.2 Segmentace

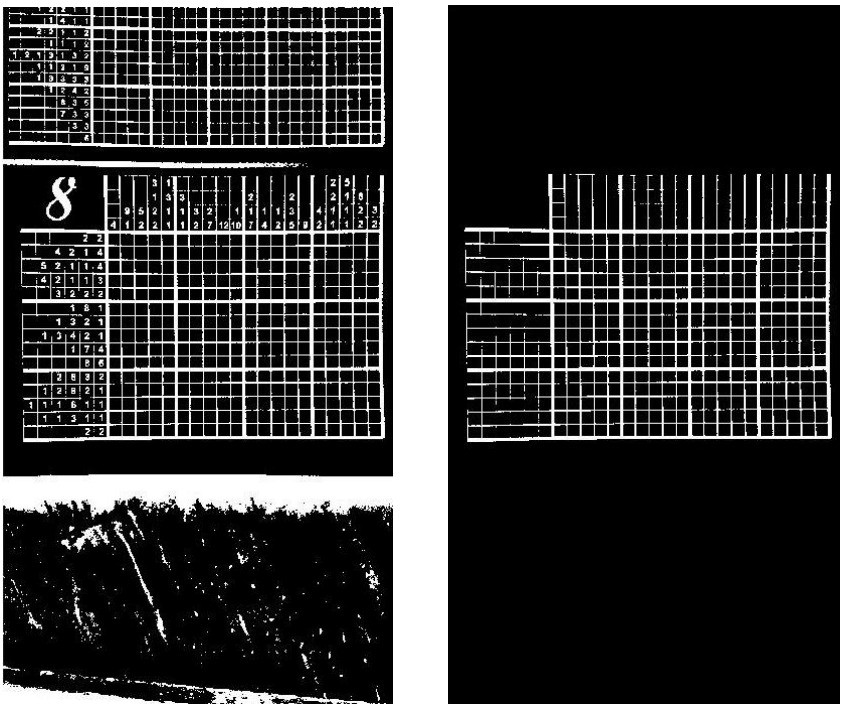
Cílem segmentace je rozčlenit obraz do částí, které souvisí s předměty či oblastmi zájmu.[3] V tomto případě je to samotný hlavolam a jeho části: legenda, mřížka hlavolamu. Segmentaci obrazu je řešena ve dvou částech: oddělení nonogramu od zbytku obrazu (vyřezané části obrazu, která obsahuje křížovku) a segmentace vyřezaného obrazu na další 3 části.

2.2.1 Oddělení hlavolamu od zbytku snímku

Jak je zmíněno v úvodu této části, snažíme se dostat obrázek, který bude obsahovat jenom samotnou křížovku. Protože se předpokládá, že snímek byl připravený tak, aby obsahoval hlavolam, který chceme vyřešit. Předpokládá se, že největší souvislá část v obrazu je křížovka samotná. Pro tento účel je použito filtrování podle velikosti spojitých oblastí. To není nic jiného než spočítání pixelů jednotlivých spojitých komponentů. Pro tento účel jsou zvoleny funkce Matlabu „bwareafilt“[10] (pro výběr největšího komponentu) a „regionprops“[11] (pro oříznutí obrazu). Funkce vyžadují inverzi naprahovaného snímku, poněvadž jako aktivní pixel považují pixel s hodnotou jedna.

2.2.1.1 Izolování největšího komponentu

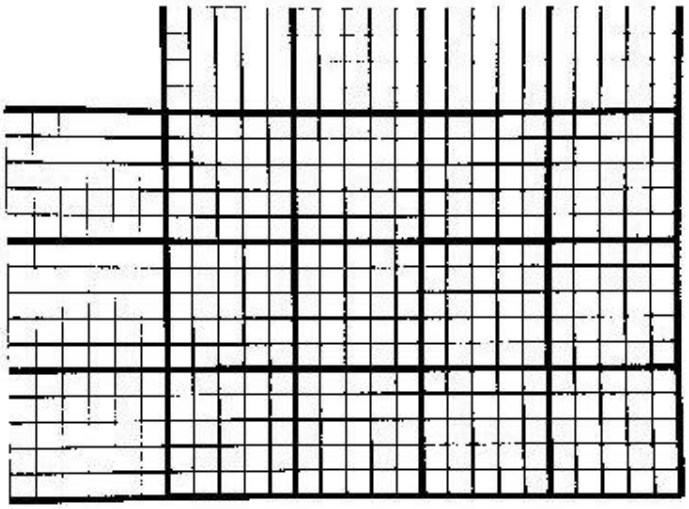
Funkce „bwareafilt“ je schopna filtrovat binární snímek podle velikosti jednotlivých komponentů. Jako parametr funkce, můžeme zadat kolik největších komponentů chceme nechat. Zvolíme-li jedničku, v obraze ponechá jenom největší spojitou oblast (viz níže).



Obrázek 16: Negativ naprahovaného obrazu (vlevo) a izolovaný největší spojitý komponent (vpravo)

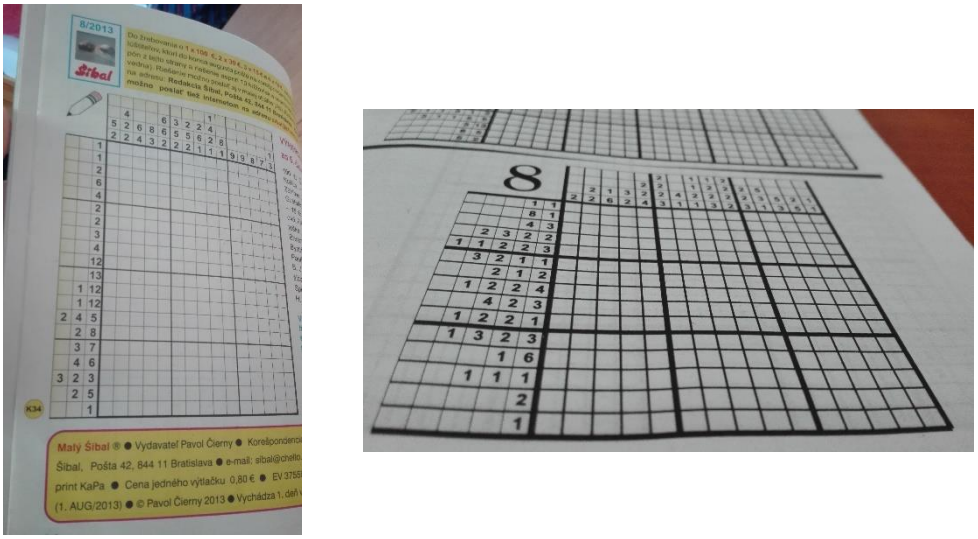
2.2.1.2 Určení velikosti hrany čtverce hlavolamu

Je potřeba určit, jak velká je hrana čtverců, ze kterých se skládá celý objekt, pro další zpracování. Protože je už obraz obsahující rastr křížovky k dispozici, tak je lehké určit průměrné velikost čtverců. Pro tento případ je použita funkce Matlabu „bwconncomp“[12],



Obrázek 17: Negativ obrazu největšího oblasti v obrazu

která vrátí strukturu četností jednotlivých oblastí. Protože velká oblast okolo hlavolamu je bílá a malé oblasti četností 1-5 pixelů tam zůstaly i po mediánové filtraci, tak by obyčejné průměrování těchto hodnot generovalo špatný výsledek. Je třeba používat takzvaný upravený průměr (angl. trimmed mean). Tato metoda průměrování hodnot nám uspořádá vektor hodnot, ze kterých se počítá průměr (od začátku a od konce vyloučí několik hodnot a ze zbytku se spočítá průměr)[13]. Výhodou metody je vyloučení extrémních hodnot při průměrování, což je v tomto případě výhodné. Protože obrázek neobsahuje číslice, které by mohly snížit velikost oblasti čtverců, je možné při dostatečně velkém procentu trimmingu získat celkem přesný výsledek o velikosti buňky hlavolamu. Jelikož známe průměrnou velikost oblasti a je známo, že buňky mají tvar čtverce, tak je možné vypočítat průměrnou délku hrany čtverce mřížky. Pokud je papír na snímku příliš ohnutý nebo hlavolam je pod extrémní perspektivou, tak tato metoda nefunguje.



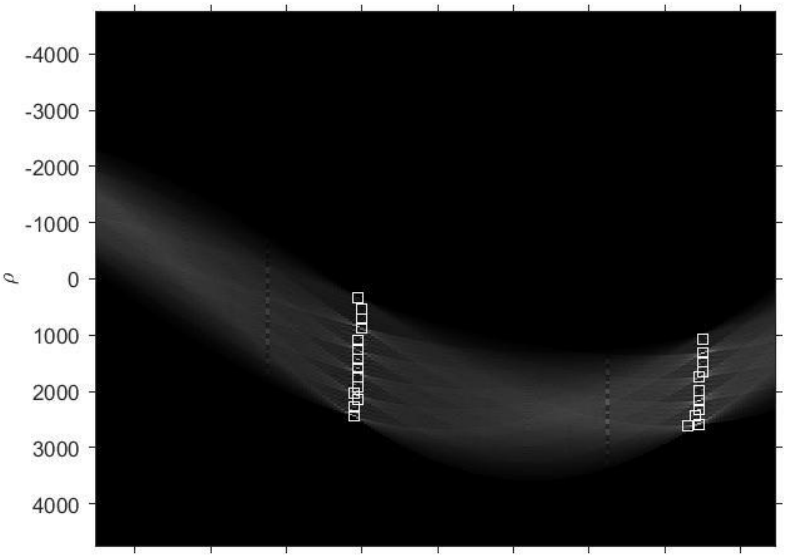
Obrázek 18: Ohnutý papír (vlevo) a obrázek s příliš velkou perspektivou (vpravo)

2.2.1.3 Vyrovnání obrazu

Pro další zpracování (obzvlášť pro rozpoznání číslic) je výhodné, aby byl hlavolam vyfotografován shora a bez zkreslení, proto je nutné nejprve obraz vyrovnat tak, aby tuto podmínku splnil.

Pokud byl nonogram na původním obrazu focen pod úhlem (obraz s perspektivou) nebo je otočený vůči rámečku obrazu, je nutné provést další úpravy (korekce zkreslení, otočení).

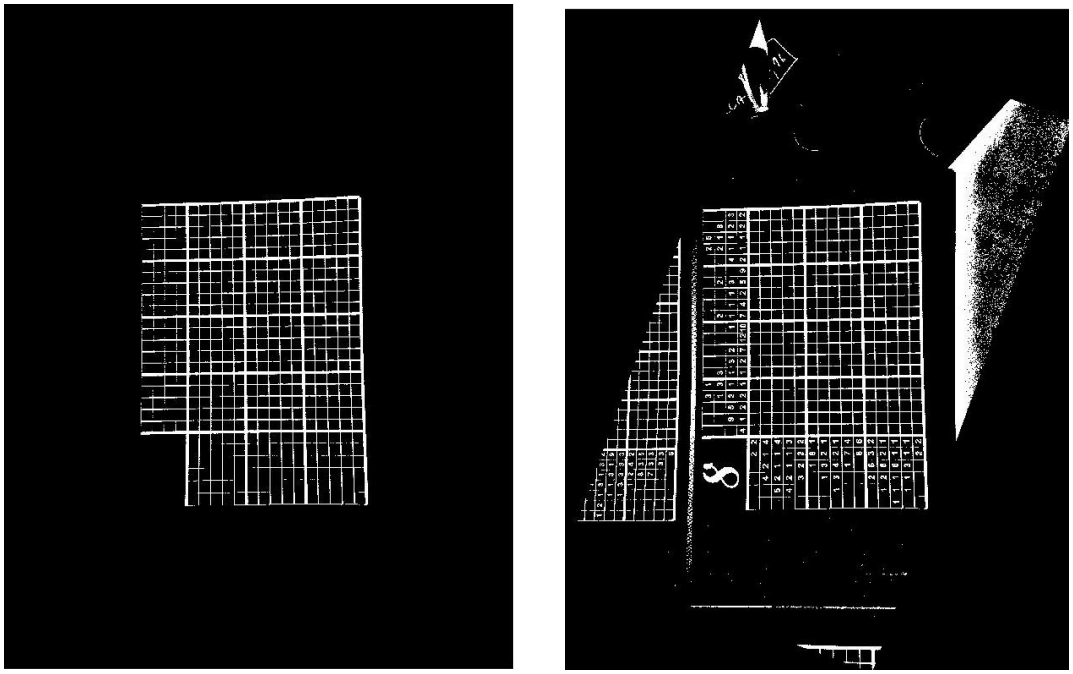
Pro tento účel je použita standardní Houghova transformace. Tato transformace slouží k nalezení geometrických entit v obrazu (čar)[5], které jsou nutné pro zjištění úhlu natočení za podmínky, že rastr hlavolamu je tvořen čtverci. Funkce „hough“[14] v Matlabu spočítá standardní Houghovu transformaci obrazu, která přesně udává, kde se jednotlivé



Obrázek 19 Pořízený snímek (nahoře) a Houghův obraz snímku (dole)

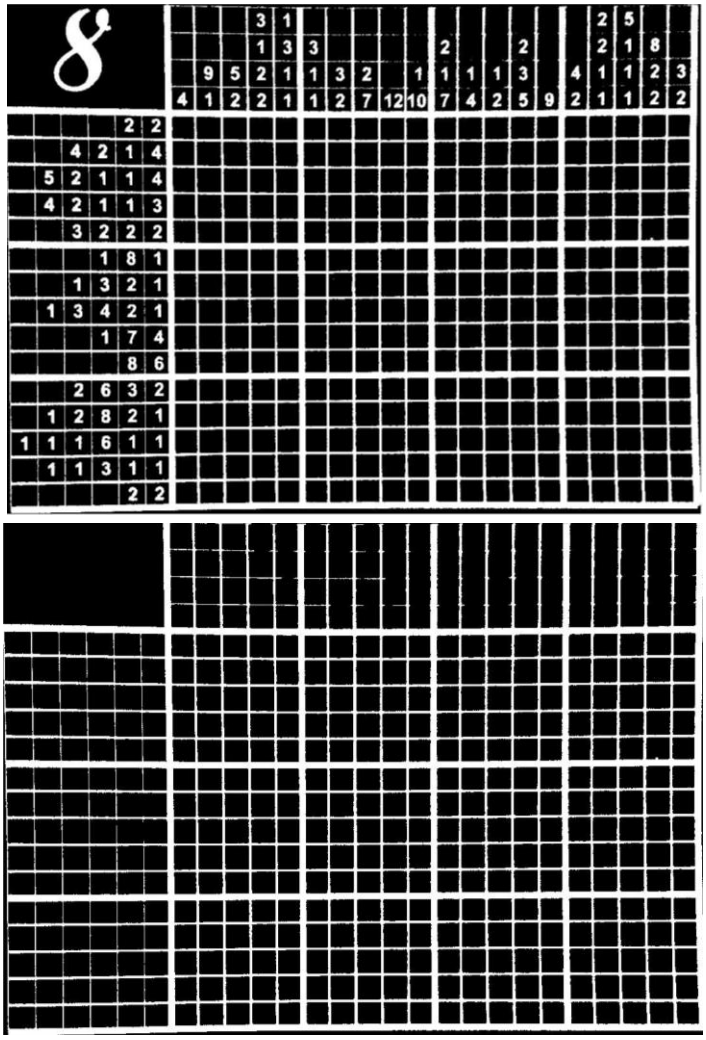
čáry nachází a pod jakým úhlem jsou vidět. Pro získání těchto informací z Houghova obrazu nám pomůže funkce „houghpeaks“[15], která vyhledá několik nejintenzivnějších špiček v tomto obrazu a „houghlines“[16], která vyhledá počáteční a koncový bod těchto vybraných čar a určí jejich vzdálenost od geometrického středu obrazu a úhel otočení.

Jestliže obraz neobsahuje hrubé geometrické zkreslení, musí platit, že existují čáry, jejichž úhel je právě o devadesát stupňů větší nebo menší než úhel dominantní. Protože Hougova transformace udává úhly v záporném smyslu, jak je vidět na obrázku: úhly cca. -20° („vodorovné“ čáry) a 70° („svislé“ čáry). Toto může způsobit chybu, kdy po otočení obrazu bude o ±90°nebo o 180° víc otočený, ovšem to zatím není problémem.



Obrázek 20: otočený komponent (vlevo) a stejně otočený naprahovaný obraz (vpravo)

2.2.1.4 Oříznutí objekt zájmu z původního obrazu



Obrázek 21:Oříznutý obraz největšího spojitého komponentu (nahoře) a naprahovaný obraz oříznutí podle stejných koordinát (dole)

Jestliže obrázek obsahuje jenom mřížku hlavolamu, koordináty rohů ohraničovacího obdélníku dávají dostatek informací potřebné pro oříznutí snímku. Zmíněné koordináty udávají funkce „regionprops“[11]. Tato funkce je použita i v následujících kapitolách pro různé účely. Tady ji voláme s opcí „BoundingBox“. Návratovou hodnotou je struktura, z které je možné určit body ohraničovacího obdélníku. Zbytek obrazu je možné odstranit a výsledkem je pouze obraz s hlavolamem. Jelikož byly doposud použity pouze jasové transformace, je možné odříznout nadbytečné části snímku (původního, šedotónového i naprahovaného) podle získaných souřadnic.

Tato metoda segmentace funguje u všech snímků z databáze. Je to poměrně rychlá metoda, a protože pracuje s ohraničovacím obdélníkem, nedojde k odstranění důležitých částí, i když obrázek obsahuje geometrické zkreslení. Pokud snímek obsahuje více hlavolamů, metoda vždy ponechá ten největší (pro řešení jednotlivých hlavolamů, je nutné rozdělit naskenované snímky na menší části, které obsahují pouze jednu křížovku).

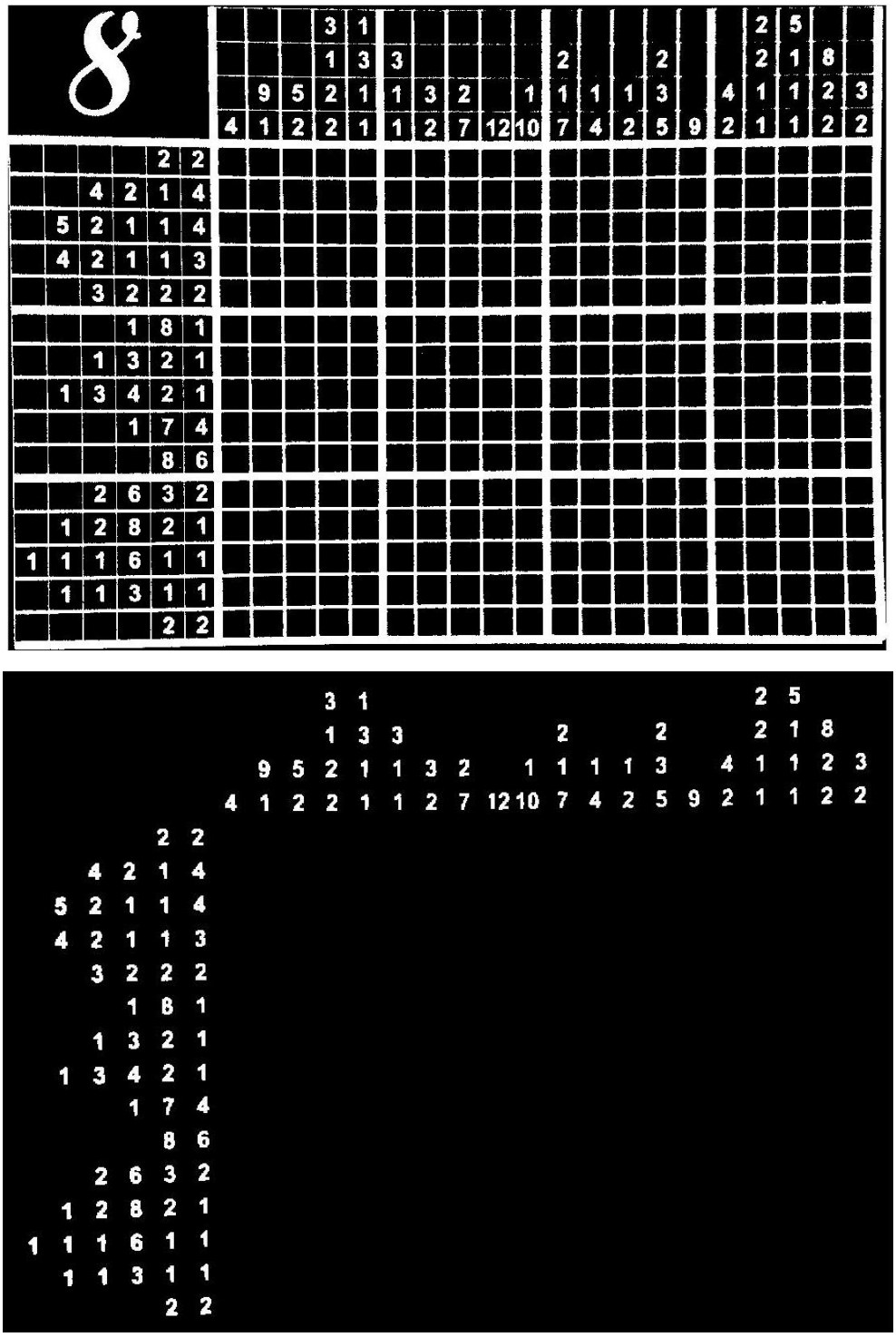
2.2.2 Segmentace legendy a pracovní plochy hlavolamu

Cílem této podkapitoly je získat digitální podobu legendy hlavolamu z upraveného obrazu hlavolamu (získat tedy 2 snímky, které obsahují pouze části obrazu, kde se legenda nachází, pokud je to možné).

2.2.2.1 Oddělení číslic od zbytku obrazu

V prvním kroku je prováděno morfologické otevření s čtvercovou maskou, aby byl obraz zbaven malých pixelů a případně došlo k rozdělení slepených číslic. Velikost masky záleží na velikosti délky stran mřížky. Obecně se dá říci, že u obrázků s menším rozlišením, tato velikost bude menší, a naopak u obrázků s větším rozlišením větší (v případě, že obrázek větším rozlišení by obsahoval hlavolam na malém části snímku, rastr hlavolamu by nebyl největším komponentem a tím pádem by došlo k chybě). Funkce „bwconncomp“ sestavuje strukturu, která obsahuje počet pixelů jednotlivých oblastí. Největší komponenta je odstaněna, neboť je jednoznačné, že to bude mřížka. To má své výhody a nevýhody. Výhodou je, že veškeré předem vyplněné části, které jsou spojené s mřížkou křížovky, jsou odstraněny také. Nevýhodou je, když se nějaká číslice dotýká mřížky hlavolamu, potom bude odstraněna také.

Největší četnost pixelů má číslice 8 a nejmenší číslice 1. Protože legenda může obsahovat libovolný počet číslic osm a jedna, je třeba při užití „bwareafilt“[10] (funkce, které nechá v obraze oblasti v mezi pixelové četnosti ponechat dostatečný práh, aby nedošlo k odstranění číslic. Interval ohraničený zdola třetinou průměrné pixelového množství oblasti a shora trojnásobkem průměrné pixelového množství se ukázal jako ideální a dostačující podmínkou. Relativní pixelový rozdíl mezi číslicí jedna a osm nebylo možné přesně kvantifikovat, ale při zúžení předem zmíněného intervalu (dolní mez na polovinu a horní mez na dvojnásobek průměru) došlo k chybě v případě, že legenda obsahovala mnohem víc jedniček než osmiček. Číslice osm má zhruba dvojnásobný počet pixelů než číslice jedna.

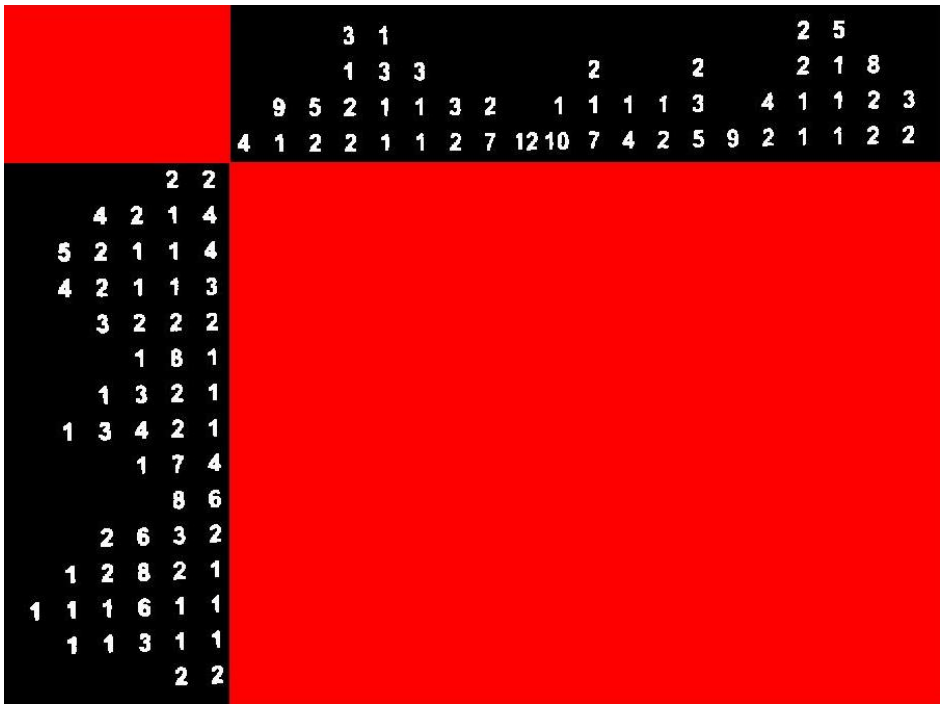


Obrázek 22: Obrázek před (nahoře) a po (dole) využití filtrování oblastí

Protože metoda je dynamická (záleží na počtu pixelů oblastí daného snímku) lze ji použít i na snímky naskenovaných větších hlavolamů, kde číslice mají menší počet pixelů.

2.2.2.2 Rozdělení legendy na dva části

Obraz získaný z předchozího bodu lze rozdělit na menší části a to tak, že nalezneme co nejmenší oblasti v obrazu, které jsou pro nás důležité. Vyznačené červené oblasti na snímku níže jsou zahozeny.



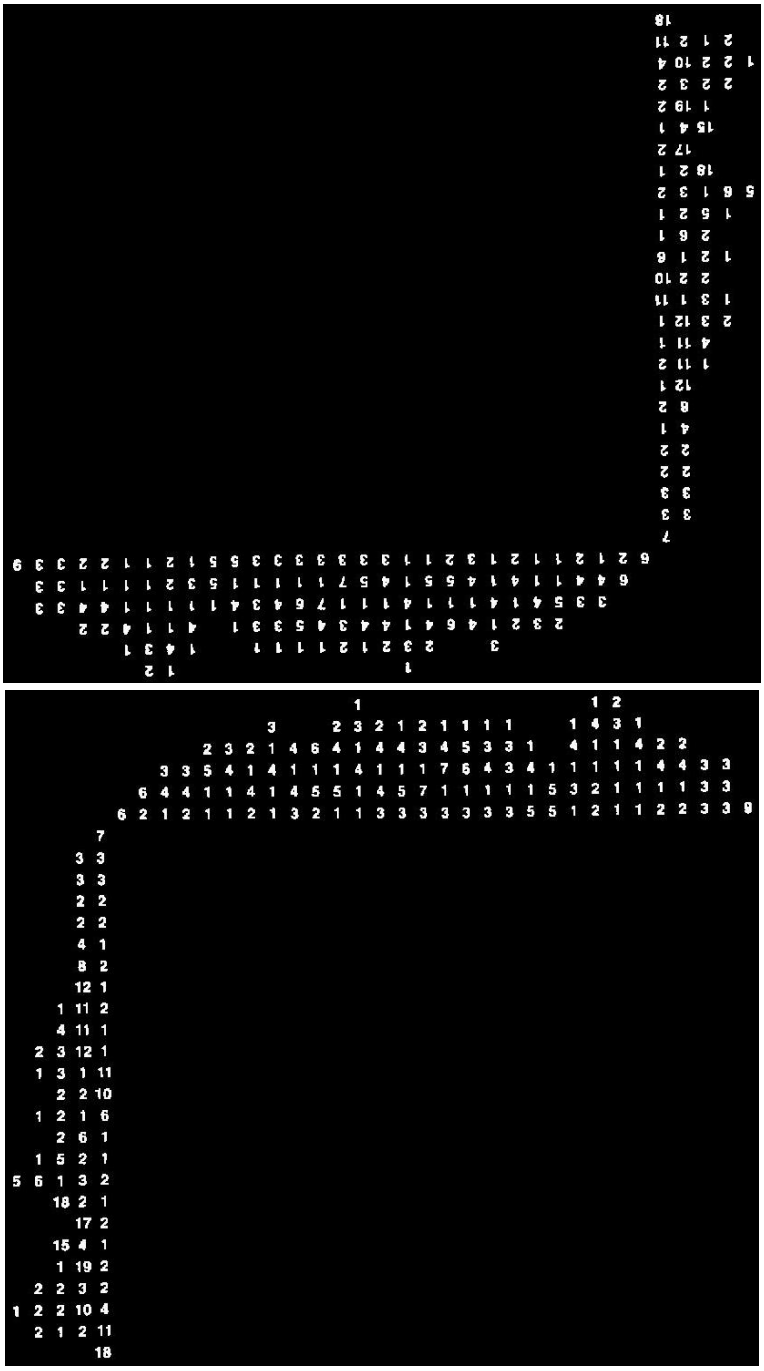
Obrázek 23: Obrázek obsahující legendu číslicemi a vyznačené oblasti, které mají být odstraněny (červeně)

Pro tento účel jsou vytvořeny vlastní cykly, které určují koordináty bodů v obrazu, podle kterých je provedena další segmentace.

V odstavci 3.1.2.3 je již zmíněno, že obrázek by mohl být o celé násobky 90° otočený. Aby bylo možné toto korigovat, jsou použity vektory, které obsahují počet pixelů v jednotlivých řádcích a v sloupcích. Je nutné zdůraznit, že se předpokládá, že legenda hlavolamu se nachází nahoře a na levé straně oblasti, kterou má uživatel vyplnit. Kdyby tomu tak nebylo, bylo by třeba čtyři případy (každé uspořádání legendy včetně seřazení číslic).

V prvním kroku jsou sečteny hodnoty a zároveň je určen i počet (poněvadž se jedná o binární obraz) bílých pixelů v jednotlivých řádcích a jednotlivých sloupcích do jednoho vektoru. Zpravidla dolní část, resp. pravá část, legendy obsahuje nejvíc číslic (existují výjimky). Součet pixelů by měl být největší právě v těchto částech vektorů.

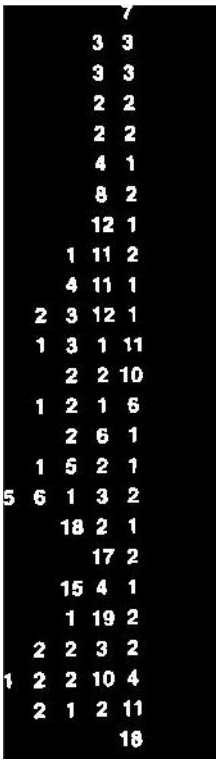
Podle toho, jestli maximální hodnota vektorů leží před polovinou celkové délky vektoru nebo až za polovinou délky, je obrázek otočen o násobky devadesáti stupňů podle kombinace polohy maxima obou vektoru.



Obrázek 24: Obrázek obsahující pouze číslice, před (nahoře) a po (dole) použití otočení pomocí maxima vektorů

Protože maximum vektorů ukazuje na levý horní bod, který je nutný pro segmentaci, bylo by jednodušší použít koordinátu tohoto bodů. Avšak situace není tak jednoduchá, neboť první i další řádky zprava (respektive řádky zdola u horní části) mohou obsahovat stejný počet číslic, jak první řádek. Je tedy možná ztráta řádku/sloupce číslic v legendě. Z tohoto důvodu bylo nutné určení koordinát, podle kterých se segmentuje, vyřešit vlastní funkcí.

Tato funkce má jako vstupní parametr vektor součtu pixelů (zmíněné v předchozím odstavci). Pokud je obrázek dostatečně vyrovnaný, vektor by měl vypadat následovně: nulové prvky (reprezentující oblast mezi sloupci/řádky, kde žádná číslice není) jsou následovány řadou nenulových prvků (jsou číslice v tomto sloupci a řádku) a tak dále až do konce vektoru. Každý vertikální i horizontální vektor by měl obsahovat místo, kde součet za sebou následujících prvků bude mnohem větší než součet řady nenulových prvků. Jestliže se legenda nachází v předem definovaném místě obrazu je patrné, že ještě před polovinou vektoru musí dojít k „pádu součtů“. Vlastní navržená funkce hledá právě toto místo. Návratovou hodnotou funkce je místo poslední nenulové hodnoty po sobě jdoucích čísel vektorů, jejichž součet byl menší než polovina součtů prvků před tím. Jsou-li známa místa, kde došlo k pádu součtů ve vektorech, je možné obraz dále segmentovat na další menší části, které obsahují jenom číslice z legendy hlavolamu. Po použití této metody je získán obrázek, který je podobný obrázku níže.

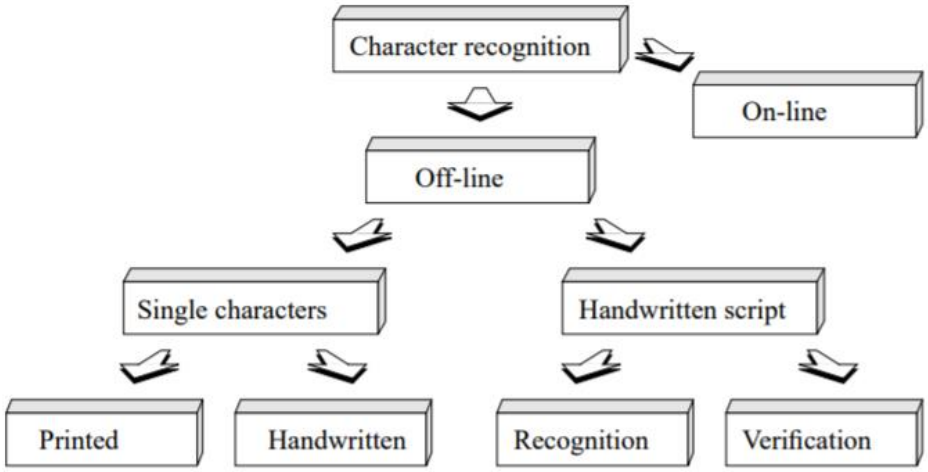


Obrázek 25: Segmentovaná část legendy nonogramu; levé část (vlevo) a horní část (vpravo)

2.3 Rozpoznávání a uspořádání číslic

Tato část je zaměřena na rozpoznávání číslic pomocí optické rozpoznávání charakteru (angl. optical character recognition, zkratka OCR) a na práci s již detekovanými čísly. Úkolem je jednoznačně určit polohu číslic, do kterého řádku/sloupce patří a uložit je podle těchto kritérii.

Optické rozpoznávání znaků se zabývá problémem rozpoznávání opticky zpracovaných dat. Optické rozpoznávání se provádí offline po zápisu nebo tiskun, na rozdíl od on-line rozpoznávání, kde počítač rozpozná znaky, jak jsou nakresleny. Ručně i tištěné znaky mohou být rozpoznány, ale výkon je přímo závislý na kvalitě vstupních dokumentů.[17]



Obrázek 26: Různé oblasti rozpoznávání znaků[17]

V práci je využívaná OCR funkce Matlabu se zaměřením na oblast rozpoznávání tištěných znaků (číslic). Funkci matlabu „ocr“[18] lze používat pro libovolný obrázek, který je Matlab schopen zpracovat. Výsledkem funkce je objekt „ocrText“[19], která obsahuje následující pole:

- samotný text, který funkce rozpoznala
- bounding box – nejmenší obdélník, ve kterém může být charakter uložen
- pravděpodobnost nebo šance, že rozpoznaný charakter je charakterem, který Matlab myslí
- stejné tři výše uvedené informace pro rozpoznaná slova

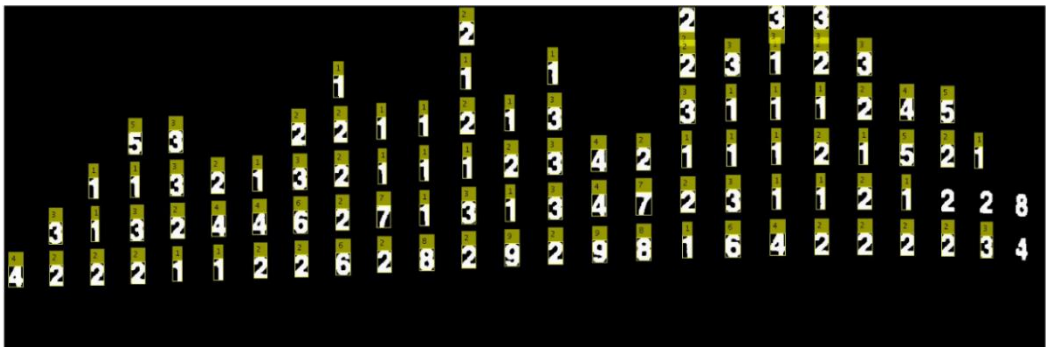
Všichni tři uvedené informace jsou pro další zpracování a identifikaci číslic nutné, protože tato část programu trvá asi nejdéle. Pouhé aplikování funkce Matlabu na obrázek by vyhodnotilo nežádoucí výsledky jako například písmena. Při volání funkce je nutné zadat kromě segmentovaného obrazu následující parametry:

- character set = hledané znaky
- text layout = uspořádání textu

Protože hledáme jenom číslice, bylo by zbytečné hledat i písmena, což by vedlo kromě zpomalení programu k nežádoucím výsledkům, které by bylo poté nutné manuálně odstranit. Jako uspořádání textu je zvolena možnost blokového textu. Takže se každý rozpoznaný znak bere zvlášť, a nikoliv jako jedno slovo. Automatické spojení znaků do „slova“ nefunguje tak, jak bylo zamýšleno, protože legenda není strukturována jako text v knihách.

2.3.1 Rozpoznávání číslic

Po použití funkce s parametry popsané výše na obou obrázkách legendy dostaneme třídu obsahující výsledky. Protože se v obraze mohou vyskytovat nežádoucí oblasti jako například část mřížky hlavolamu, které mají srovnatelný počet pixelů jak číslice a jsou zase nežádoucími výsledky, je ho třeba filtrovat. To znamená vyloučit NaN výsledky a podle pravděpodobností jednotlivých rozpoznaných znaků (tzn. aby svislá čára s čtyřicetiprocentní pravděpodobností nebyla identifikovaná za jedničku). Z několika pokusů bylo zjištěno, že ideální pravděpodobnost, nad kterou jsou čísla správně rozpoznána je nad 50 %. Nastavení práh je zvolen v rozsahu 55 až 60 procent, poněvadž při volbě prahu nad 60 %, se stane, že číslice, která by byla potřebná, je zahozena.



Obrázek 27: nerozpoznané číslice z důvodu nastavení vysokého prahu pravděpodobnosti správnosti rozpoznání

Když je naopak zvolena nižší úroveň, podle které je určováno, která číslice je správná, dojde k tomu, že jedna číslice je rozpoznána jako dvě různé číslice vedle sebe, např. číslice osm je určena jako číslice šest a tři.

Číslice, které byly smazány při segmentaci, jsou přeposlané pro OCR ještě jednou přímo z naprahovaného obrazu, kdy jsou ještě v mřížce přítomna. Funkce pro OCR není schopna tyto číslice rozpoznat a vrací buď špatný výsledek nebo prázdnou třídu.

Obrázek 28: ukázkové obrázky zpracované funkcí „ocr“, která vrátila prázdný nebo špatný výsledek.

Vstupní obraz by bylo možné jinak naprahovat nebo přímo poslat šedotónový obraz do funkce, ale někdy i samotná pozice číslic je pochybná. Není známo, kolik čísel by jeden sloupec/řádek v legendě měl obsahovat. Rovněž je neznámé, kolik sloupců/řádků v legendě dohromady je. Bylo by možné říci, že rozměr nonogramů bude vždy dělitelný pěti, ale stejně by to moc nepomohlo. Jediným dobrým řešením by bylo naučit OCR na vlastní vzory, ale to by bylo až příliš náročné vzhledem sestavení databáze. Ve většině případů vhodný práh pro identifikaci je dostačující. Nejčastěji jsou odstraněny číslice dvouciferných čísel.

2.3.2 Uspořádání rozpoznaných číslic

Předpokládá se, že již všechna čísla jsou rozpoznána a nechybí ani jedno z nich. Jelikož je pozice číslic v obraze známá (pomocí boundingboxů), tak již uspořádání číslic do struktury, která je vhodná pro další zpracování, není problémem, avšak požaduje vlastní algoritmus. Tento algoritmus se trochu liší v případě horní části a levé části legendy, proto je funkčnost algoritmu rozdělena na spořádající algoritmus. Kdybychom měli jenom jednociferná čísla, jednalo by se o úpravu matic na dva řádky. Problémem jsou dvouciferná čísla.

2.3.2.1 Uspořádání rozpoznaných číslic v horní části legendy

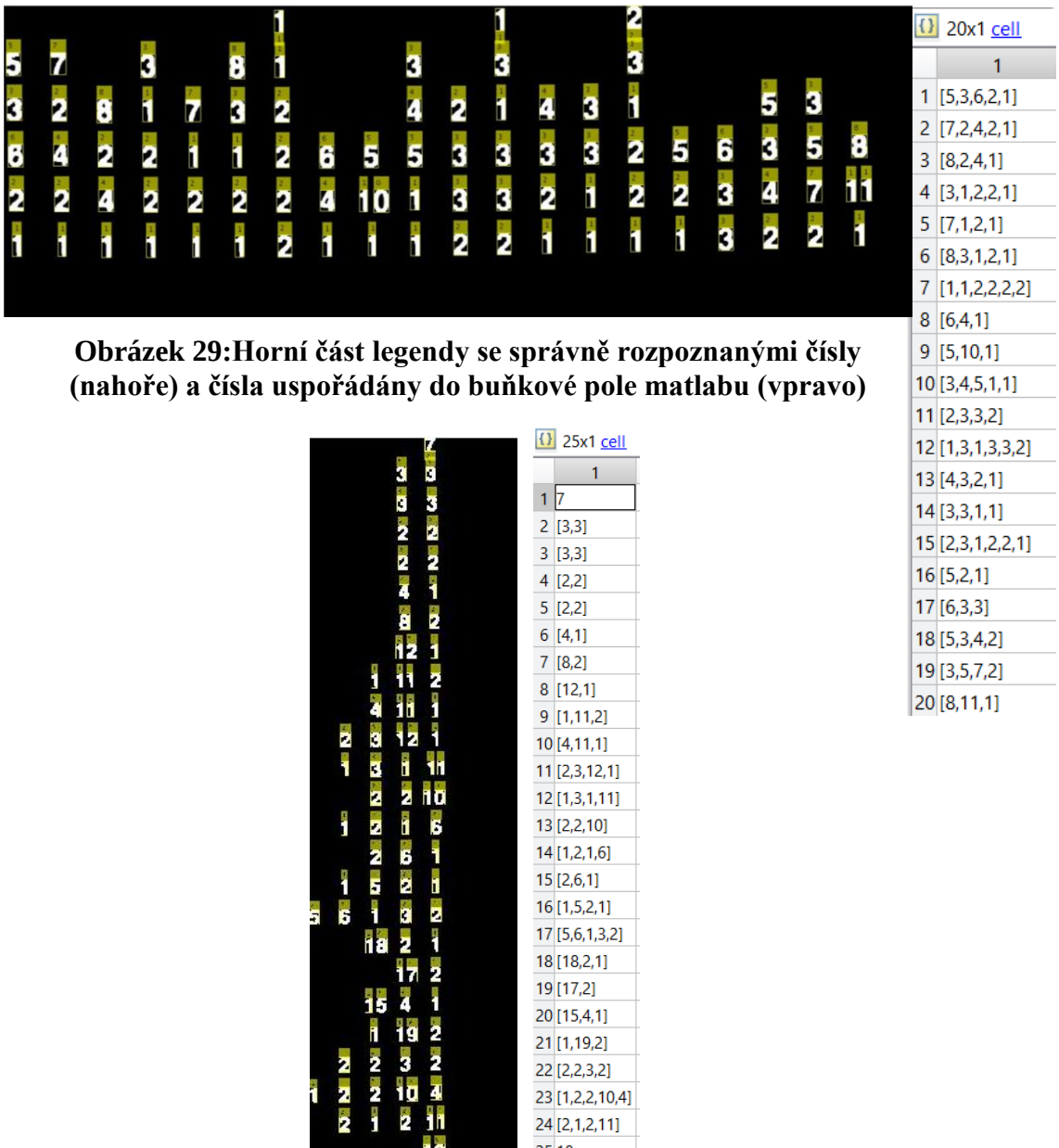
V prvním kroku je třeba vytvořit matici z koordináty „boundingboxů“ a rozpoznaných číslic. To lze snadno udělat, pokud je k dispozici matice, která obsahuje x-ové a y-ové koordináty ohraničeného obdélníku číslic a samotná čísla, která byla identifikována funkcí „ocr“.

V prvním kroku je nutné zjistit počet sloupců. To je vyřešeno pomocí funkce „sortrows“[20], která uspořádá matice podle řádků zadaného sloupce a podle zadaného směru. Protože se jedná o číslice ve sloupcích, poloha x-ové koordináty jednoznačně určí, ve kterém sloupci se daná číslice nachází. Známe-li průměrnou vzdálenost číslic (na tento účel je použita průměrná délka čtverce mřížky hlavolamu), jsme schopni jednotlivé sloupce od sebe rozdělit. Další uspořádání číslic je provedeno v již oddělených sloupcích podle koordináty x (čili řádků). To by postačilo v případě, kdyby v legendě nebyla žádná dvouciferná čísla. Pro rozpoznání a uložení dvouciferného čísla do struktury je využita tu vlastnost těch číslic, že byli zařazeny do stejného sloupce, avšak y-ová koordináta obdélníků ve kterém se nacházejí mají zhruba stejné. Protože se může stát, že „boundingbox“ druhé číslice dvouciferného čísla (na místo jedniček) má větší y-ovou hodnotu než číslice na místě desítky, není vhodné spojit tyto číslice podle y-ových koordinát obdélníku, proto jsou použity koordináty x-ové. Mohou nastat dva případy. V prvním případe číslice vlevo má větší y-ovou koordinatu boundingboxu a vdruhém případě číslice ležící vpravo má větší y-ovou koordinátu boundingboxu.

2.3.2.2 Uspořádání rozpoznaných číslic v levé části legendy

Princip je obdobný. Rozdíl je jenom v pořadí uspořádání číslic a ve spojení číslic dvouciferného čísla. Na začátku jsou číslice uspořádány podle řádků a potom podle polohy x-ové hodnoty obkresleného obdélníku. Tady je jednodušší, že je jednoznačné, která číslice u dvouciferného čísla je první.

Jak je patrné z obrázků 29 a 30, dvouciferná čísla jsou uspořádány správně. Každé je tam, kde má být a přímo ve tvaru, který většina řešících algoritmů požaduje. Tady končí část zpracování obrazu.



Obrázek 29:Horní část legendy se správně rozpoznanými čísly (nahoře) a čísla uspořádány do buňkové pole matlabu (vpravo)

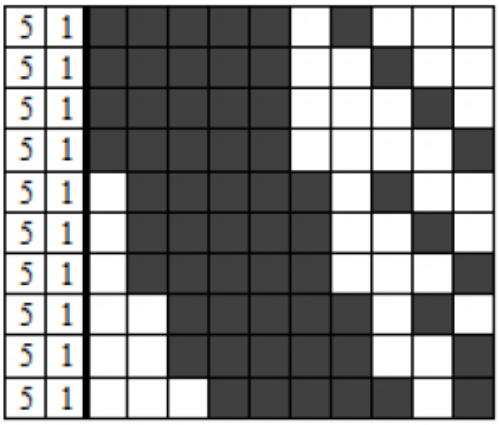
Obrázek 30: Levá část legendy se správně rozpoznanými čísly (vlevo) a čísla uspořádány do buňkového pole matlabu (vpravo)

3. SOLVER

V kapitole č. 2 bylo ukázáno, jak se řeší jednoduchý nonogram jednoduchými metodami. Pro větší (složitější) nonogramy existují komplexní metody, protože při jejich řešení jednoduché metody selhávají a není možné pokračovat v řešení. V tomto případě je nutné uhodnout, zda nějaké políčko bude vyšrafováno nebo bude ponecháno prázdné, poté je možné pokračovat v řešení již zmíněnými metody, dokud někde nedojde k rozporu mezi legendou a vyšrafovanými oblastmi. V tomto případě je třeba se vrátit zpět ke kroku, kde bylo předtím rozhodnuto o vyšrafování špatných poliček a pokračujeme v řešení jinou cestou. Tato metoda je ale velice obtížná a zdlouhavá pro manuální (lidské) řešení.

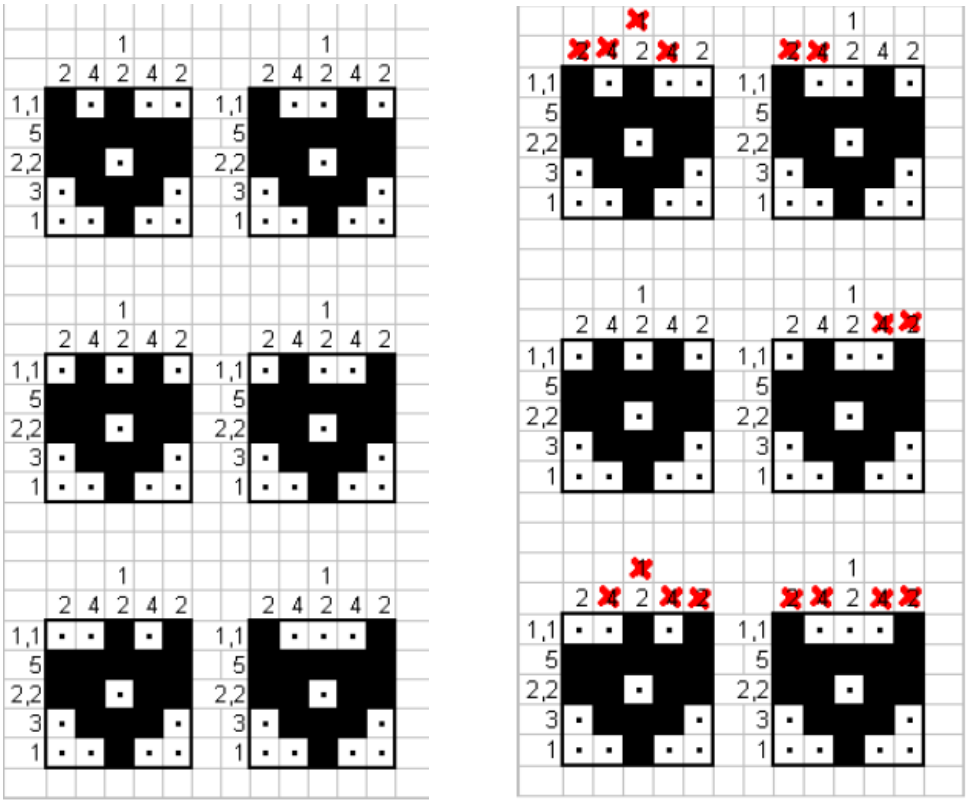
3.1 Depth-first search (prohledávání do hloubky)[21]

Tato metoda zkouší různé kombinaci možnosti uspořádání vybarvených částí pro každý řádek. Na Obrázku 31 je ukázka všech možností umístění bloků s délkou 5 a 1 v délce řádku o 10 blocích.



Obrázek 31: DFS - možnosti umístění bloků v řádku[21]

Řešit nonogram pomocí této metody je poměrně jednoduché pro pochopení. Vezmeme si první řádek hlavolamu a vygenerujeme každou možnost pro umístění bloků v tomto řádku. Když je každá možnost pro první řádek vyčerpaná, přejdeme do druhého řádku a zopakujeme proces. Takto přejdeme přes celou křížovku. Jakmile je nalezeno řešení pro každý řádek, tak se pomocí legendy sloupců určí správná kombinace vyřešených řádků, jak je znázorněno na Obrázku 32.



Obrázek 32: Výsledky DFS (vlevo) a ověření správnosti výsledků (vpravo)[21]

Jak je vidět, jediná kombinace horního řádku, která není v rozporu s legendou sloupců, je třetí kombinace. V tomto případě jsme našli řešení hlavolamu velice rychle, protože ostatní řádky byly již předem vyplněné. U větších hlavolamů časová náročnost algoritmu extrémně narůst. Složitost algoritmu je matematicky popsána, jako $O(b^d)$, kde b je takzvaný faktor rozvětvení a d je prohledávaná hloubka stromu. Takže v první řadě čas, který je potřebný pro vyřešení hlavolamu, je závislý na počtu řádků křížovky. Dále na permutaci rozložení bloků v řádcích a na počtu sloupců. To znamená, že když v první řade musíme umístit jeden vyšrafovaný bloček a máme 10 sloupců, tak máme 10 možností, jak je umístit. Když musíme umístit souvisle 5 bloků za sebou ve stejném hlavolamu, tak počet možností je jenom 5.

3.2 Další řešící algoritmy

Existuje mnoho dalších algoritmů řešící podobné problémy jako modifikované verze metody DFS, které například fungují rychleji v případě, že některé pozice jsou již známy, jak je vidět na Obrázku 32. Pro větší hlavolamy je mnohem rychlejší genetický algoritmus. Genetický algoritmus používá biologicky odvozené techniky jako dědění, přirozený výběr, rekombinace a mutace. Takové algoritmy je třeba trénovat, aby byly efektivní a rychlé. V některých případech genetický algoritmus může uvíznout. Proti uvíznutí se lze bránit detekcí, kdy se populace algoritmu již nevyvíjí. Pro menší křížovky algoritmus DFS překonává genetický algoritmus.

3.3 Použitý solver

Jelikož zpracování obrazu je celé řešeno pomocí programu Matlab, bylo výhodné zvolit řešící algoritmus, který je psaný v programovacím jazyku, který je kompatibilní s programem Matlab a je možné používat jeho funkce nebo metody používat. Matlab nabízí široký výběr jazyků, jejichž kód lze volat i z Matlabu. Mezi tyto jazyky patří C, Fortran, Python (verze 2.7, 3.5 a 3.6), Java 8, .NET CLR (verze 2.0, a 4.0), http 1.1, Perl 5.26.1.[22] Volání kódu se ale liší podle typu jazyku. Nejsnadněji lze implementovat solver v jazyce Java.

Použitý kód je mírně modifikovaná verze programu, který napsal github uživatel Samjayy.[23] Tento solver využívá modifikovanou DFS, která funguje mnohem rychleji na větších hlavolamech, kde už jsou některá pole předem vyplněna. Návratová hodnota kódu je textový soubor, která obsahuje řešení ve formě uvedené na Obrázku 33.

OUT:

```
Time: 46ms
#######0###000#0#0#######
#00000#0##0##00000#00000#
#0###0#00000###0#0#0###0#
#0###0#0#00######0#0###0#
#0###0#00#####0##0#0###0#
#00000#00##0000000#00000#
#######0#0#0#0#0#0#######
00000000###000###00000000
#0##0###00#0#0###0#00#0##
#0#000000###0##0000#000#0
0####0#0####0##0#0000##00
0#0#000#000#0#0####0#0###
00##00#0#0#000000##0#####
000###0##0##0######0###0#
#0##########0#0#00##0000#0
0##0#00##000##0###00000#0
###0#0#0#00#0000#####0#00
00000000#000##0##000#####
#######0#00##000#0#0#0###
#00000#0##00#00##000##0#0
#0###0#000####00#####00#0
#0###0#0###0##########0##
#0###0#0#00######0######0
#00000#00##000000#0#0##00
#######0##000#0##000#####
```

Obrázek 33: Výstup solveru

Z obrázku je patrné, že kromě výsledku solver do výstupu vypíše i čas, který byl potřebný pro vyřešení problému. Z mého hlediska je tato informace nadbytečná, takže jsem z výsledného kódu tuto část odstranil. Dále je vidět, že výstupní znaky, které

reprezentují, jestli daná pozice v křížovce má být vyšrafovaná nebo ne, nejsou ryze nuly a jedničky (to by z hlediska dalšího manipulování s výstupem solveru bylo lepší), ale obsahuji i jiné znaky. Nebylo by složité znaky „#“ vyměnit za čísla, poněvadž v kódu jsou to statické proměnné reprezentující prázdnou nebo vyšrafovanou buňku, ale výstupem solveru je textový soubor, takže při formátování výsledku na matici je to jedno, jestli formátujeme ASCII hodnotu jedničky nebo znak # na číselnou hodnotu jedna. Dále v případě, že nonogram má více než jedno řešení, algoritmus vrátí to řešení, které našel jako první.

Aby bylo volání programu z prostředí Matlab možné, je zapotřebí používat stejnou verzi Javy, jakou používá Matlab původně (nebo je také možné zadat absolutní cestu k verzi Javy, kterou má Matlab použít) a zadat cestu ke třídě, kterou voláme. Cestu ke třídě můžeme zadat dynamicky nebo staticky. Protože dynamická cesta nefungovala (nenašel danou třídu z nějakého důvodu), byla lokace třídy předána přes soubor obsahující absolutní cestu. Tento soubor musí být umístěn ve složce Matlabu pod názvem „javaclasspath.txt“. V mém případě je to na disku C v složce: R2018a.

Z důvodů složitosti předávání vstupných hodnot do solveru a zpracování návratových hodnot ze solveru jsou zvoleny souborové vstupy a výstupy. To znamená, že v prostředí Matlab je vytvořen vstupní soubor pro řešící algoritmus, který má podobu:

- R S (dimenze hlavolamu; R – počet řádků, S – počet sloupce.
- R řádků (pro každý řádek legendy, jednotlivé číslice odděleny mezerou)
- S řádků (pro každý sloupec legendy, umístěny obdobně jako řádky)
- R řádků s S charaktery, které udávají počáteční stav hlavolamu:
    - 0 - neznámý stav buňky
    - # - vyplněná oblast

Jak už bylo zmíněno a je patrné i z uspořádání vstupných dat, solver je schopen vyřešit částečně vyplněné nonogramy. Protože u zpracování obrazu nebyly zjištěny žádné předem vyplněné oblasti, tato možnost není použitá.

4. UŽIVATELSKÁ APLIKACE

Pro jednoduché ovládání programu jsem vytvořil uživatelskou aplikaci pomocí prostředku „appdesigner“ Matlabu. Tato aplikace umožňuje laikům používat program.

4.1 AppDesigner

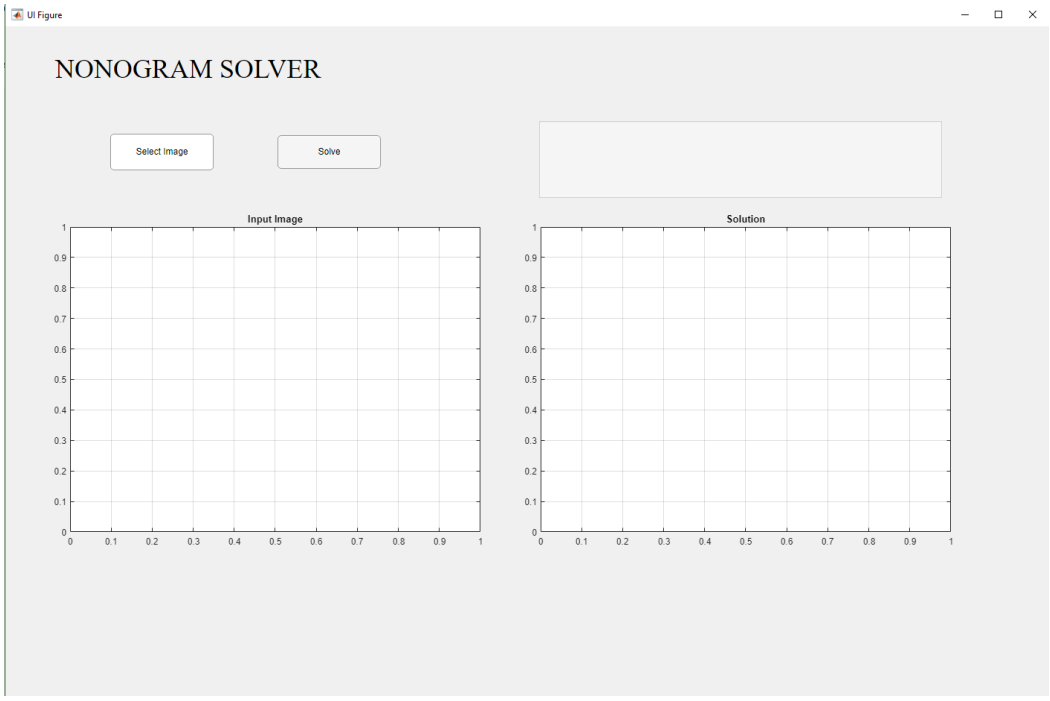
Aplikace byla vytvořena ve vývojovém prostředí Matlab „AppDesigner“. Toto prostředí je rafinovanější verze prostředí GUI, které bylo poprvé představeno ve verzi 2016a.[24] Toto prostředí umožnuje vytvářet uživatelské aplikace z kódu napsaného v Matlabu. Programování může probíhat v kódovém a v grafickém pohledu. V grafickém pohledu (na plátně) můžeme umístit různé textové pole, tlačítka, slidery a další podobné prvky. V kódovém prostoru můžeme přidat funkčnost k předem zmíněným věcem. Takto vytvořené vývojové prostředí se chová jako jedna třída, ve které každý komponent má svoji vlastní metodu. Jestliže uživatel s daným komponentem může manipulovat, je metoda veřejná, statické textové pole a názvy štítky jsou privátní.

4.2 Popis vytvořeného uživatelské rozhraní

Pro zobrazení výsledku je vytvořeno jednoduché uživatelské rozhraní, ve kterém jsou umístěny následující komponenty:

1. Vlevo nahoře se nachází statický štítek „NONOGRAM“.
2. Pod štítkem jsou dvě tlačítka.
   2.1. Select Image - výběr obrázku, pomocí kterého se vybírá obrázek, který chceme poslat řešícímu algoritmu.
   2.2. Solve - příkazové tlačítko pro provedení kódu řešícího nonogram z předaného obrazu, pokud jsme pomocí tlačítka „Select Image“ vybrali obrázek.
3. Nemodifikovatelná textová pole, která slouží především pro zobrazení chyb.
4. Dva větší grafické prostory rezervované pro zobrazení obrázků:
   4.1. Na levé straně se zobrazí již načítaný vstupní obraz.
   4.2. Na pravé straně se objeví výsledek, ať už byl nalezen.

Původně bylo zamýšleno přidat nápovědu pro řešení v podobě matic tisknutelných tlačítek podobnou tomu, jakou používá známá počítačová hra „Minesweeper“ (po česky Hledání min)[25]. Z důvodů chyby kompatibilit a časových důvodů nebyl nápad realizován.



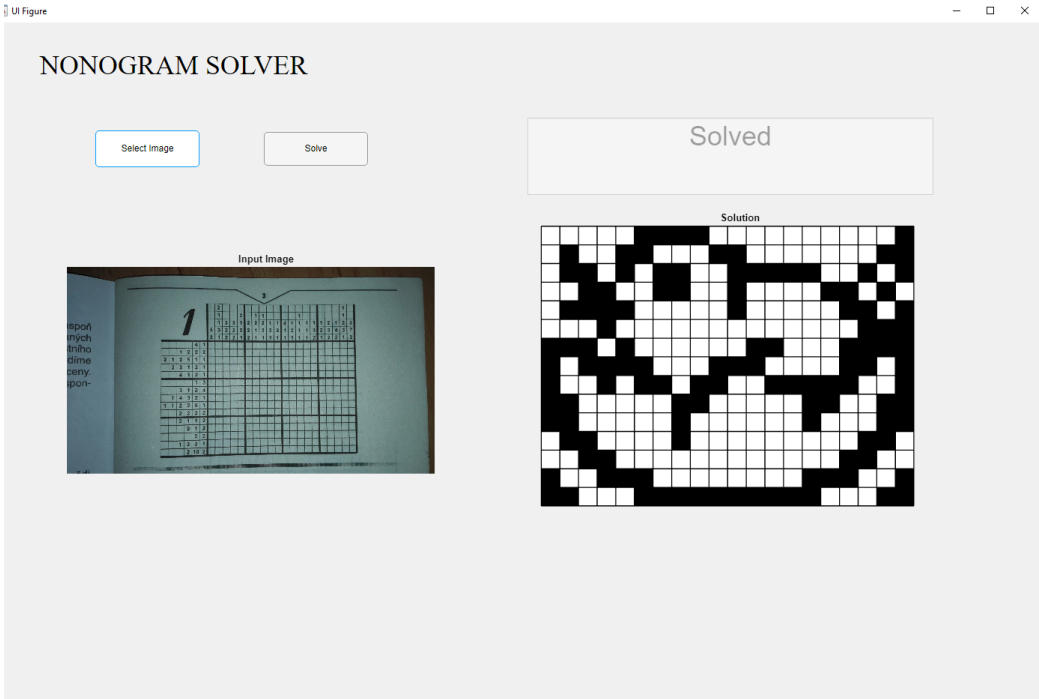
Obrázek 34: Uživatelské rozhraní po otevření

4.2.1 Vybírání obrázku

Vybírání obrázku probíhá pomocí matlabovské funkce „uigetfile“, která otevře rozhraní pro vybírání souboru již známého z e-mailových klientů při vkládání příloh. Dovolené formáty obrazu, jsou formáty: .jpg, .jpeg, .bmp a .png. Při vybíraní souborů nejsou ani jiné formáty nabídnuty. Po vybírání snímku se snímek uloží do takzvané „property“ proměnné, kterou lze použít i mimo funkce tlačítka a zároveň se otevřený obrázek zobrazí ve vyhrazeném okně „Input Image“.

4.2.2 Výpočet řešení a zobrazení výsledku

Po načítání obrazu, kliknutím na tlačítko „Solve“ se provádí algoritmus pro řešení vybraného obrazu. Výsledek se zobrazí v okně „Solution“ a v textovém poli se zobrazuje hlášení „Solved“.



Obrázek 35: Úspěšně vyřešený nonogram

Jestli dojde k chybě v řešení, výsledek se nezobrazí (neboť neexistuje) a místo toho se v textovém poli objeví chybová zpráva, která může být jedna z typů:

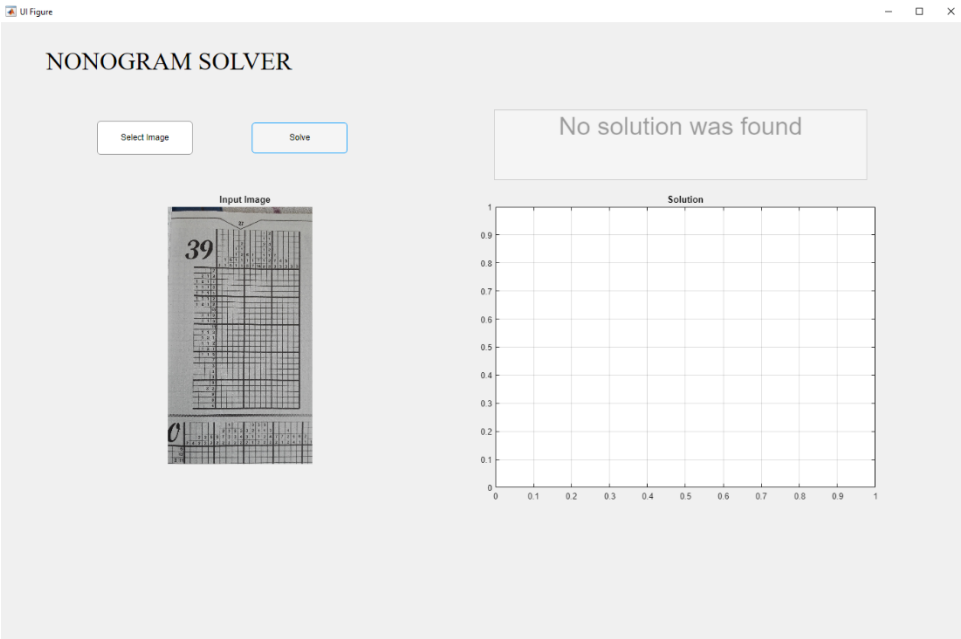
- **Error. No vermax found.**
  Zobrazí se, když cyklus nenajde bod „horizontálního maxima“ v sloupcovém vektoru diskutovaný v sekci 2.2.2.2. Je to chyba většinou způsobeno tím, že při filtrování černobílého obrazu došlo k odstranění mnoho číslic z legendy. Tato chyba je asi nejvýznamnější, může způsobit další chyby (diskutované v dalších bodech).

- **Error. No hormax found.**
  Zobrazí se, když cyklus nenajde bod „horizontálního minima“ ve vektoru zmíněnéjo v předchozím bodě.
- **Error. OCR did not find any results in „hornicisla“.**
  Předaný obrázek pro funkci OCR nenašel žádné číslice v segmentovaném obrazu „hornicisla“.
- **Error. OCR did not find any results in „levecisla“.**
  Předaný obrázek pro funkci OCR nenašel žádné číslice v segmentovaném obrazu „hornicisla“.
- **It would take a vast amount of time to solv this puzzle with DFS.**
  Upozornění, která brání programu, aby spustil řešící algoritmus pro větší obrazy než 15x25, resp. 25x15. Sice časová náročnost algoritmu DFS je dána exponenciálním $O(b^d)$, a není jedno, jestli hlavolam má velikost 15x25 nebo

25x15, ale experimentálně bylo zjištěno, že největší rozměr křížovky, která je řešitelná v akceptovatelném množství času (tj. za sekundy), je 15x25.

- **Error. Solution does not contain enough elements to reconstruct the picture.**
  Pakliže výstupní soubor obsahující výsledek hlavolamu neobsahuje určité množství znaků (počet prvků matic), které je určena předem ze zjištěného počtu sloupců a řádků, bude vyhlášena tato chyba.
- **No solution found**
  Jestliže z předaných informací řešící algoritmus nebyl schopni vypočítat řešení z nějakého důvodu, tato chyba bude vypsaná v textovém poli. Důvody této chyby jsou špatně segmentovaný obraz (diskutovaný v podkapitole 2.2.2) nebo špatně rozpoznané/nerozpoznané číslice s funkcí OCR (rozbor naleznete v kapitole 2.3.1).

Někdy se stane, že řešící algoritmus sice najde řešení, ale z důvodu špatně předané legendy výsledek nebude správný i když podle solveru je.



Obrázek 36: Případ, kdy během řešení došlo k chybě

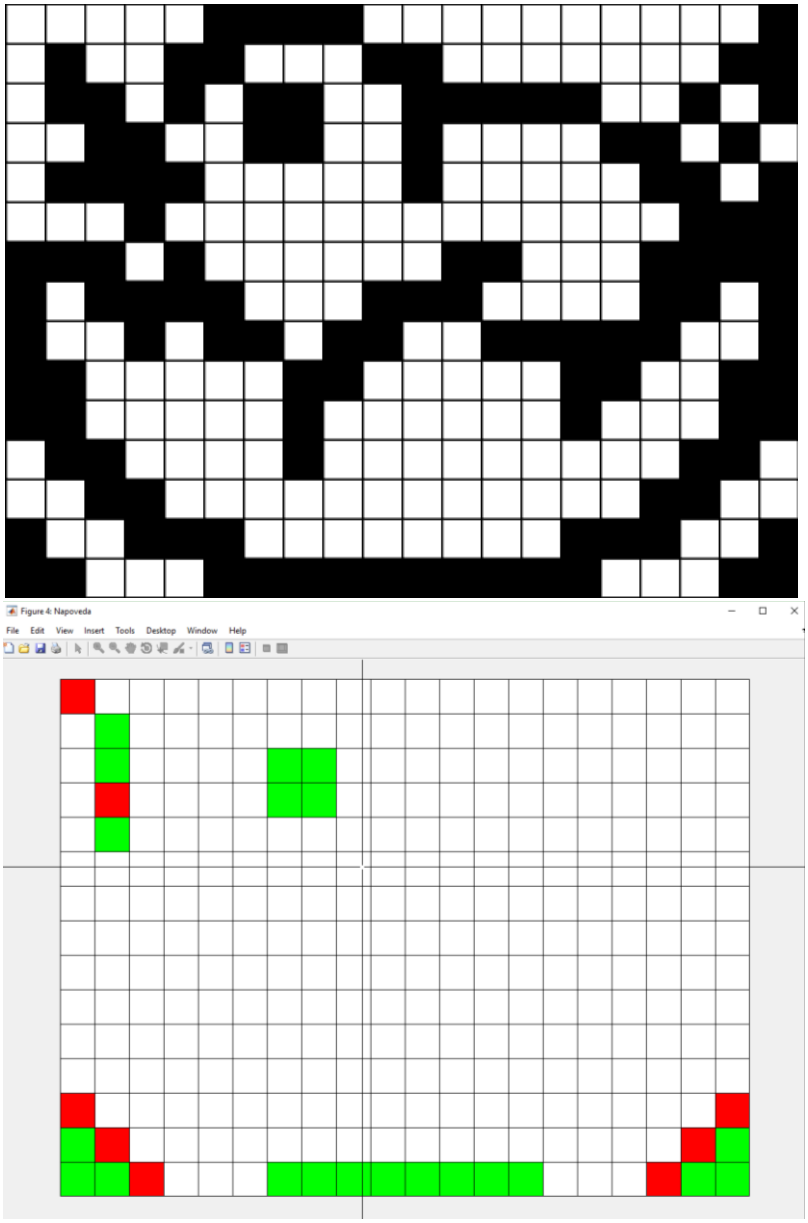
4.3 Uživatelská aplikace

Uživatelskou aplikaci je vytvořena jako aplikace v prostředí Matlab. Po spuštění Matlabu stačí jenom vyhledat složku, kde se instalace nachází a otevřít instalační soubor. Po instalaci aplikace ji lze spustit z karty „APPS“.

Aplikace funguje úplně stejně jako rozhraní (viz. podkapitola č. 4.2). Ohledem na funkce, které jsou nové ve verzi 2018a, je doporučen aplikaci spustit ve zmíněné verzi.

4.4 Nápověda řešení

Toto okno slouží k tomu, aby uživatel mohl otestovat, zda daná buňka má být vyšrafována či nikoliv. Jestliže vybraný čtvereček není součástí řešení (nemá být vypněn), po kliknutí změní barvu na červenou, naopak když je součástí řešení, tak se barva čtverečku zmení na zelenou, jak je vidět na obrazců níže.



Obrázek 37: Správné řešení hlavolamu (nahoře) a nápověda řešení (dole)

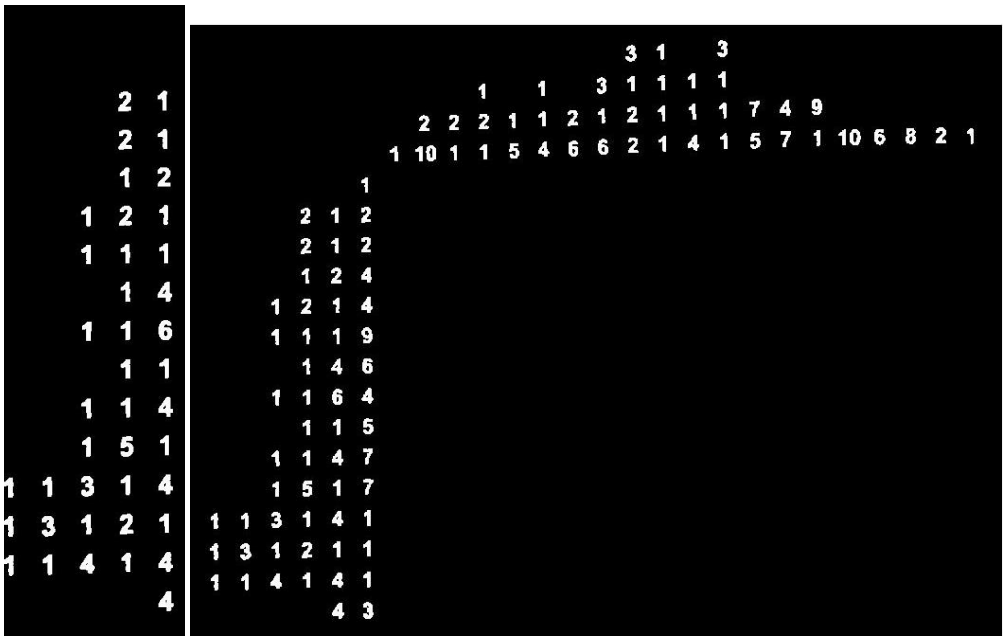
Bohužel toto rozšíření jsem nebyl schopen implementovat do samotné uživatelské aplikace, poněvadž UIaxes nepodporuje grafický input z kurzoru myší.[28] K testování se dá použít pouze skript vytvořený v matlabu.

5. ZÁVADY PROGRAMU

Zadání práce jsem zkusil splnit podle mých nejlepších poznatků, avšak z časových a jiných důvodů se nepovedlo splnit všechno podle mých představ. Tato kapitola diskutuje nejčastější chyby programu a dalších funkcí, které by mohly být zlepšeny.

5.1 Odstranění geometrického zkreslení

Protože nonogram nemá pevně daný tvar, je těžké si vybrat univerzální metodu pro odstranění zkreslen. Existují různé funkce v Matlabu na řešení tohoto problému, ale nebyl jsem schopen úspěšně naimplementovat ani jednu, která by splnila očekávání. Geometrické zkreslení v mnoho případů způsobilo chyby při ořezávání legendy, poněvadž pravý sloupec levé části legendy a spodní řádek horní části legendy nebyly na sebe kolmé. Došlo tedy k oříznutím části legendy. Tato chyba se vyskytla v 29,09% případů testovaných 55 snímků.



Obrázek 38: Špatně segmentovaná legenda (vlevo) a původní černobílý obraz (vpravo)

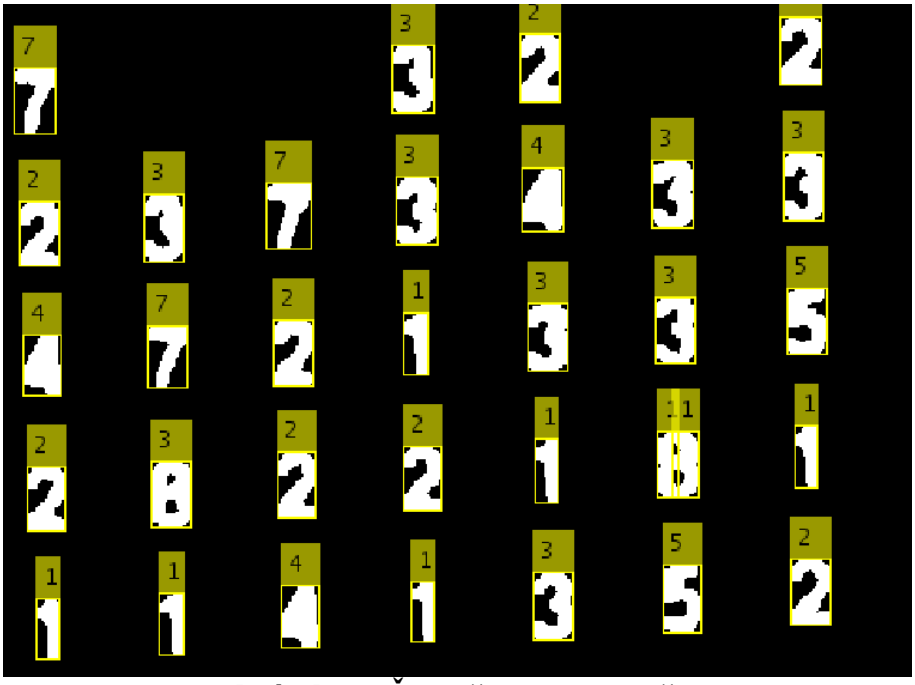
5.2 Odstranění číslic jako následek odstranění malých komponentů z obrazu

V některých případech došlo k odstranění číslic kvůli prostorové filtraci objektů v obraze. Nejedná se o odstranění číslic, které dotýkají mřížky hlavolamu, ale o ty, které byly odstraněny důsledkem pevně definovaného prahu pixelové četností. Tato chyba byla podstatná při testování na databázi skládající se z naskenovaných obrazů (snímky o

malém rozlišení: menší než 1000x1000 pixel). Tato chyba se vyskytla v každém případě. Právě proto jsem byl nucen tuto databázi (obsahující 30 snímků) vynechat ze statistiky a z dalšího testování algoritmu. U obrázků s větším rozlišením došlo k odstranění důležitých části zřídka. Většinou to byly číslice spojené s mřížkou.

5.3 Nerozpoznaný a špatně rozpoznané číslice

V tomto případě jde o závady způsobené tím, že číslice dobře segmentovaného obrazu byly špatně rozpoznány pomocí OCR funkce Matlabu. Tento problém silně souvisí se nastaveným prahem pravděpodobnosti, zda rozpoznaná číslice je opravdu stejná jako číslice, kterou Matlab vydává za rozpoznanou. Kvůli předem zmíněným závadám, zejména odstranění a zanechání komponentů v obraze nebylo možné nastavit menší práh pravděpodobnosti uznání výsledků z OCR než 50 %. Pod tímto prahem by byly uznány i některé zbytky mřížky jako platné číslice (nejčastěji jako jedničky). Naopak nad tímto prahem by nebyly některé číslice rozpoznány. Z již zmíněných důvodů je těžké tuto chybu kvantifikovat, neboť může být následkem dalších chyb. Následek chyby způsobené ryze z důvodu OCR Matlabu je vidět na obrazu níže, kde osmička byla rozpoznaná jako dvě jedničky těsně vedle sebe.



Obrázek 39: Špatně rozpoznaná číslice

5.4 Program není schopen detekovat již vyplněné části hlavolamu

Protože geometrické zkreslení z obrazu není odstraněno, bylo by obtížné kontrolovat vyplnitelnou oblast hlavolamu, zda se v ní nachází křížek nebo je celý čtverec vyšrafovaný. Další důvody, proč nebyla řešena část s detekcí vyplněných

oblastí, jsou následující. První důvod je databáze obrázků, která obsahuje hlavolamy, ve kterých je legenda napsaná do čtverečků, takže je obtížné najít horní levý roh vyplnitelné oblasti. Druhým důvodem jsou velké vyšrafované oblasti v hlavolamu, které by znemožnily rekonstruovat původní mřížku (jedině z koordinát čísel v legendě by to bylo možné).

6. NÁPADY NA VYLEPŠENÍ

Program řešící zpracování obrazu je celkem primitivní. Proběhne jenom jednou a jakmile na nějakou chybu narazí, ohlásí chybu a okamžitě končí, Většinu prahů určí pomocí jedné proměnné, a to je průměrná délka hrany čtverce v hlavolamu. Celé řešení by se dalo řešit jako stavový automat, kde jednotlivé části kapitoly 2 proběhly v cyklu a jakmile někde dojde k chybě, například nedostatek výsledků z funkce OCR, tak by se program vrátil ke stavu, kde filtroval objekty a nastavil by jiný práh pro filtraci. Vzhledem k tomu, že proces zpracování obrazu i takto trvá tři až pět sekund, doba zpracování by byla ještě delší, ale důsledkem by byly mnohem přesnější výsledky. K tomu by bylo ale potřeba odstranit geometrické zkreslení.

Po úspěšném vyrovnání obrazu by byly rozpoznatelné již předem vyšrafované oblasti křížovky. V současné situaci to nemá cenu řešit, protože jakmile nějaká větší oblast je vyšrafovaná, není možné určit pozice vyplněných buněk, poněvadž buňky nejsou stejně velké, někdy ani nejsou v jedné řadě (v případě, že je papír ohnutý). Předem vyplněné části by usnadnily práci řešícího algoritmu DFS.

Solver bychom mohl vyměnit za dobře natrénovaný genetický algoritmus, takže by bylo možné řešit i nonogramy, u kterých legenda byla úspěšně rozpoznaná, ale z důvodů časové náročnosti algoritmu DFS řešení nebyla provedena. Dalších 22 obrázků (z 55) by bylo řešitelných v tomto stavu, v případě že se zpracování obrazu provede pouze jednou. Další možnost je použít oba algoritmy a podle velikosti hlavolamu by program zavolal metodu DFS nebo genetické algoritmus.

Uživatelská aplikace by mohla zobrazit ukryté řešení v podobě matic čtverců, kde by uživatel mohl hrát hru podobnou k Hledání min.

7. ZÁVĚR

Práce je rozdělena na tři hlavní kapitoly, kromě jiných menších doplňujících.

První kapitola obsahuje obecné informace o japonských křížovek o jejich historie a základní metody řešení nonogramu.

První velká část (2. kapitola) řeší zpracování obrazu, zejména segmentaci a rozpoznávání číslic. Segmentace proběhla úspěšně u 70,91 % testovaných obrázků. Rozpoznávání číslic bylo u 72,73 % obrázků úspěšné, avšak kompletní zpracování obrazu dalo správné celkové výsledky (kdy vzájemně byla úspěšné segmentovaná legenda hlavolamu, a i rozpoznávání číslic úspěšné) v 49,10 % případech všech testovaných snímků.

Druhá část řešení (3. kapitola) se zabývá výběrem řešícího algoritmu a jeho implementováním. Solver pracuje s metodou hrubé síly. Vybraný algoritmus funguje bez chyby, jedinou nevýhodou je časová náročnost výpočtu větších hlavolamu. Právě z toho důvodu jsem byl nucen přidat ošetřovací podmínku, podle které solver přeběhne nebo ne. To bohužel snížilo celkové množství obrázků, které mohou být vyřešeny v přiměřené lhůtě.

Třetí část (4. kapitola) se zabývá s vytvořením uživatelské aplikace, která byla navržena ve vývojovém prostředí AppDesigner. Úkolem této aplikaci je usnadnit práci s již napsaným kódem a graficky znázornit výsledky.

O známých chybách a nápadech na vylepšení vytvořeného programu pojednává kapitola pátá a šestá.

Celkový výsledek této práce je program vytvořený v Matlabu, který je schopen vyřešit černobílé japonské křížovky, takzvané nonogramy podle předaných digitálních obrazů křížovky.

Literatura

[1] Nonogram. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-12-29]. Dostupné z: https://en.wikipedia.org/wiki/Nonogram

[2] Methods of solving Japanese crosswords. *Nonograms.org* [online]. nonograms.org, 2009 [cit. 2018-04-24]. Dostupné z: http://www.nonograms.org/methods

[3] SONKA, Milan, Vaclav HLAVAC a Roger BOYLE. *Image Processing, Analysis, and Machine Vision*. 2008. Toronto, Canada: Thomson Learning, 2008. ISBN 978-0-495-24428-7.

[4] *Rgb2grey* [online]. Natick, Massachusetts, USA: The MathWorks, 2018 [cit. 2018-05-17]. Dostupné z: https://www.mathworks.com/help/matlab/ref/rgb2gray.html

[5] NIXON, Mark S. a Alberto S. AGUADO. *Feature Extraction and Image Processing*. 2008. London, UK: Elsevier, 2002. ISBN 978-0-12372-538-7.

[6] *Imbinarize* [online]. Natick, Massachusetts, U.S.A.: The MathWorks, 2016 [cit. 2018-04-22]. Dostupné z: https://www.mathworks.com/help/images/ref/imbinarize.html

[7] EDDINS, Steve. Adaptive thresholding for binarization. In: *MATLAB Central* [online]. -: MATLAB, 2016 [cit. 2017-12-29]. Dostupné z: https://blogs.mathworks.com/steve/2016/07/25/adaptive-thresholding-for-binarization/

[8] Salt-and-pepper noise. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-04-22]. Dostupné z: https://en.wikipedia.org/wiki/Salt-and-pepper_noise

[9] *Medfilt2* [online]. Natick, Massachusetts, USA: The MathWorks, 2018 [cit. 2018-05-17]. Dostupné z: https://www.mathworks.com/help/images/ref/medfilt2.html

[10] *Bwareafilt* [online]. Natick, Massachusetts, USA: The MathWorks, 2018 [cit. 2018-05-17]. Dostupné z: https://www.mathworks.com/help/images/ref/bwareafilt.html

[11] *Regionprops* [online]. Natick, Massachusetts, USA: The MathWorks, 2018 [cit. 2018-05-17]. Dostupné z: https://www.mathworks.com/help/images/ref/regionprops.html

[12] *Bwconncomp* [online]. The MathWorks, Inc.: Natick, Massachusetts, USA, 2018 [cit. 2018-05-17]. Dostupné z: https://www.mathworks.com/help/images/ref/bwconncomp.html

[13] Trimmed mean. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: https://en.wikipedia.org/wiki/Truncated_mean

[14] *Hough* [online]. Natick, Massachusetts, USA: The MathWorks, 2018 [cit. 2018-05-17]. Dostupné z: https://www.mathworks.com/help/images/ref/hough.html

[15] *Houghpeaks* [online]. Natick, Massachusetts, USA: The MathWorks, 2018 [cit. 2018-05-17]. Dostupné z: https://www.mathworks.com/help/images/ref/houghpeaks.html

[16] *Houghlines* [online]. Natick, Massachusetts, USA: The MathWorks, 2018 [cit. 2018-05-17]. Dostupné z: https://www.mathworks.com/help/images/ref/houghlines.html

[17] EIKVIL, Line. Optical Character Recognition [online]. Oslo, 1993, s. 35 [cit. 2018-05-18]. Dostupné z: https://www.nr.no/~eikvil/OCR.pdf

[18] *Ocr* [online]. Natick, Massachusetts, USA: The MathWorks, 2018 [cit. 2018-05-18]. Dostupné z: https://www.mathworks.com/help/vision/ref/ocr.html

[19] *OcrText* [online]. Natick, Massachusetts, USA: The MathWorks, 2018 [cit. 2018-05-17]. Dostupné z: https://www.mathworks.com/help/vision/ref/ocrtext-class.html

[20] *Sortrows* [online]. Natick, Massachusetts, USA: The MathWorks, 2018 [cit. 2018-05-18]. Dostupné z: https://www.mathworks.com/help/matlab/ref/sortrows.html

[21] ZAVISTANAVIČIUS, Ramūnas. *Nonogram solving algorithms analysis and implementation for augmented reality system*[online]. Kaunas, Litva, 2012 [cit. 2018-05-17]. Dostupné z: https://core.ac.uk/download/pdf/51698555.pdf. Diplomová práce. Kaunas University of Technology. Vedoucí práce Dr. Andrej Ušaniov.

[22] *MATLAB Supported Interfaces to Other Languages for R2018a* [online]. Natick, Massachusetts, USA: The MathWorks, 2018 [cit. 2018-05-18]. Dostupné z: https://www.mathworks.com/support/sysreq/supported-language-interfaces.html

[23] *Solver* [online]. Samjayyy, 2015 [cit. 2018-05-18]. Dostupné z: https://github.com/Samjayyy/logicpuzzles/tree/master/nonogram

[24] *AppDesigner* [online]. Natick, Massachusetts, USA: The MathWorks, 2018 [cit. 2018-05-17]. Dostupné z: https://www.mathworks.com/help/matlab/app-designer.html

[25] Minesweeper. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: https://en.wikipedia.org/wiki/Microsoft_Minesweeper

[26] *Malované křížovky magazín*. 2018, **X.**(1.). ISSN 1335-9525.

[27] Malované křížovky: kódované obrázky. 2018, **XVII**(2.). ISSN 1335-9533.

[28] NARGUNDKAR, Ankita. Interactive figure manipulation with App Designer. *Https://www.mathworks.com* [online]. Natick, Massachusetts, USA: The MathWorks, 2017 [cit. 2018-05-19]. Dostupné z: https://www.mathworks.com/matlabcentral/answers/344795-interactive-figure-manipulation-with-app-designer

Seznam příloh

Příloha 1 - Databáze obrázků

Příloha 1 – Digitální obrázky o nonogramech. Obrázky jsou uloženy na přiloženém CD v složce „databáze“. Složka obsahuje 4 další složky

1. Složka Mobile – obsahuje obrázky pořízené mobilem z měsíčníku Malované křížovky magazín[26]
2. Složka Mobile2 – obsahuje obrázky pořízené mobilem z měsíčníku Malované Křížovky – kódované obrázky[27]
3. OneByOne – obsahuje vyřízené snímky obsahujíc jeden hlavolam na snímek z naskenovaných obrazů z měsíčníku Malované křížovky magazín[26]
4. OriginakScan – obsahuje naskenované dvoulisty z měsíčníku Malované křížovky magazín[27]

Příloha 2 - Solver

Příloha 2 – Řešící algoritmus, čili solver. Na přiloženém CD v složce „solver“ jsou soubory:

1. JAVA_nonogramSolver.zip – obsahuje použitý Java třídu (jako kompresovány NetBeans projekt)
2. matlab_script.m – použitý skript pro testování funkčnosti algoritmu
3. nonogramAppDesigner – editor uživatelské rozhraní
4. nonogramAPPinstaller – instalační soubor uživatelské aplikace
5. place – funkce použitá algoritmům
6. solve – matlab_script.m v podobě funkcí použitá v uživatelském aplikací

Příloha 3 - Dosažené výsledky

Na přiloženém CD v složce „výsledky“ naleznete výsledky úspěšně vyřešených hlavolamů. Výsledky jsou uloženy v .jpg formátu. Poslední písmeno v názvu obrazcích značí, zda-li dalý hlavolam je z měsečníku „Malované křížovky magazín“[26] (poslední písmenko je „m“), nebo z „Malované křížovky – kódované obrázky“[27] (poslední písmenko je „k“).

Příloha 4 - Videozáznam programu

Na přiloženém CD je video.mp4 fajl, které je video demonstrujíc funkčnost výsledního aplikace.