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

PŘEVOD BAREVNÉHO SNÍMKU NA ČERNOBÍLÝ
PHOTO CONVERSION TO BLACK AND WHITE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Matúš Pieger**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Miloslav Richter, Ph.D.**
SUPERVISOR

BRNO 2019

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: Matúš Pieger ***ID:*** 186162

Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Převod barevného snímku na černobílý

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte metodu převodu barevné fotografie na černobílou, včetně korekcí geometrických zkreslení, tak aby výsledek co nejlépe odpovídal předloze. U výstupního černobílého snímku umožněte volbu úrovní šedi.

1) Nastudujte metody počítačového vidění vhodné pro danou úlohu a popište jejich hlavní vlastnosti.
2) Definujte vlastnosti zdrojových snímků. Nasnímejte zdrojové snímky v celém rozsahu a kombinacích zvolených vlastností.
3) Navrhněte algoritmy pro zpracování a úpravu snímků.
4) Realizujte navržené algoritmy a výsledky zhodnoťte.

DOPORUČENÁ LITERATURA:

Žára J., Beneš B., Sochor J., Felkel P.: Moderní počítačová grafika, Computer Press, 2005, ISBN 978-80-2-1-0454-5.

Sonka M., Hlavac V., Boyle R..: Image Processing, Analysis, and Machine Vision, 3rd Edition, CL Engineering, 2007, ISBN: 978-0495082521.

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 20.5.2019

Vedoucí práce: Ing. Miloslav Richter, Ph.D.

Konzultant:

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

ABSTRAKT

Tato bakalářská práce popisuje návrh algoritmů pro detekci, úpravu a převod hledaných objektů ve vstupních barevných snímcích do šedotónové nebo černobílé oblasti. Vstupní snímky jsou rozdělené do dvou kategorií, kde první kategorie představuje snímky papírových dokumentů. Ve druhé kategorii se nachází snímky sudoku a jiných hlavolamů. Práce analyzuje nedostatky zdrojových snímků a jejich vliv na stavbu a funkčnost jednotlivých programů. Následně hledá řešení pro spolehlivou detekci požadovaných objektů a opravu jejich geometrických zkreslení. Pro lepší čitelnost, resp. šetření inkoustu při tisku výstupních snímků práce dává odpovědi na to, jak odstranit přebytečné stíny a jakým způsobem je možné zmenšit bitovou hloubku obrazu na vybranou velikost.

KLÍČOVÁ SLOVA

zpracování obrazu, šedotónový obraz, detekce hran, segmentace, geometrické zkreslení, morfologické operace, prahování

ABSTRACT

This bachelor's thesis describes the designing of algorithms for detection, processing and conversion of desired objects from input color images to grayscale or black and white domain. The input images are divided into two groups, where the first group consists of sudoku images and other brain puzzles. The second group contains images of paper documents. The thesis analyses imperfections of input images and their effect on the building and functionality of each of the programs. Subsequently, it looks for solutions of a reliable detection of desired objects and correction of their geometric distortions. For a better readability and toner saving when printing the output images respectively, the thesis gives answers as to how to erase unwanted shadows and in which way is it possible to perform bit depth reduction to a chosen number.

KEY WORDS

image processing, grayscale image, edge detection, segmentation, geometric distortion, morphological operations, thresholding

BIBLIOGRAFICKÁ CITACE

PIEGER, Matúš. *Převod barevného snímku na černobílý.* Brno, 2019. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/119012. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Miloslav Richter.

PROHLÁŠENÍ

Prohlašuji, že svoji bakalářskou práci na téma Převod barevného snímku na černobílý jsem vypracoval samostatně pod vedením vedoucího Ing. Miloslava Richtra, Ph.D. 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 práce Převod barevného snímku na černobílý 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/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **20. května 2019** ……

(podpis autora)

PODĚKOVÁNÍ

Za přátelský přístup a dobré rady při vytváření práce chci poděkovat Ing. Miloslavovi Richtrovi Ph.D.

V Brně dne: **20. května 2019** ……

(podpis autora)

OBSAH

Seznam symbolů a zkratek

ZKRATKY:

CCD	…	Charged Coupled Device
CMOS	…	Complementary Metal Oxide Semiconductor
A/D	…	analogově-digitální převodníky
RGB	…	Red-Green-Blue barevný model
2D	…	dvourozměrný
VUT	…	Vysoké učení technické v Brně
LUT	…	Look-up tabulka

SYMBOLY:

$Red(x,y), Green(x,y), Blue(x,y)$	…	Hodnota pixelu v matici dané spektrální složky	[-]
$\nabla f(x,y)$	…	Gradient obrazové funkce	[-]
$\mathrm{G_x}, \mathrm{G_y}$	…	Gradient obrazové funkce v příslušném směru	[-]
$\theta(x,y)$	…	Směr gradientu obrazové funkce	[-]
$w(x,y)$	…	Okénková funkce	[-]
$E(u,v)$	…	Hodnota rozdílu mezi původním a posunutým obrázkem	[-]
R	…	Hodnota parametru, který určuje detekci rohu u Harrisova detektoru rohů	[-]
$a_{0\ldots4}$, $b_{0\ldots4}$	…	Koeficienty matice pro afinní transformace	[-]
$a, b, c, d, e, f, g, h, i$	…	Koeficienty matice pro projektivní transformace	[-]
K	…	Koeficient projektivní transformace	[-]
u, v	…	Souřadnice vstupních bodů u projektivních transformací	[-]
$h[i]$	…	Pole hodnot histogramu, kde i představuje index daného místa v poli	[-]
$LUT[i]$	…	Pole hodnot Look-up tabulky, kde i představuje index daného místa v poli	[-]

Seznam obrázků

ÚVOD

V době levných chytrých mobilních telefonů a fototechniky obecně je pořizování kvalitních fotografických snímků dostupné jak pro průmysl, tak pro většinu populace. Souběžně s tímto trendem se otevírají nové možnosti využití pořízených snímků. Touto problematikou se zabývá obor počítačového vidění, který zkoumá možnosti extrakce hodnotných informací z obrazu pomoci různých metod jeho zpracování. Tyto informace se pak můžou využít například u ovládání procesů v autonomních vozidlech a průmyslových robotech nebo při detekci různých jevů a objektů.

Jelikož je v současnosti časté, především ve studentském prostředí, kopírování papírových poznámek nebo obecně papírových dokumentů pomoci fotografování telefonem, bude tato práce hledat řešení pro úpravu těchto typů snímků. Cílem je zpracovat snímky tak, aby výsledek co nejlépe odpovídal klasickému skenování papírů a bylo je možné zpětně tisknout. Zpracování, tak jak bylo domluveno s vedoucím práce, bude rozděleno do dvou kategorií. První kategorii budou představovat snímky papírových poznámek a jiných dokumentů. Druhá kategorie bude obsahovat fotografie sudoku a dalších hlavolamů, fotografie tabulek a jiných objektů vytištěných na papíře s jasným ohraničením. Dataset bude pro zajímavost obsahovat několik obecných snímků z různých oblastí života, které nelze zařadit do prvních dvou kategorií, na kterých ale bude možné vyzkoušet si redukci bitové hloubky.

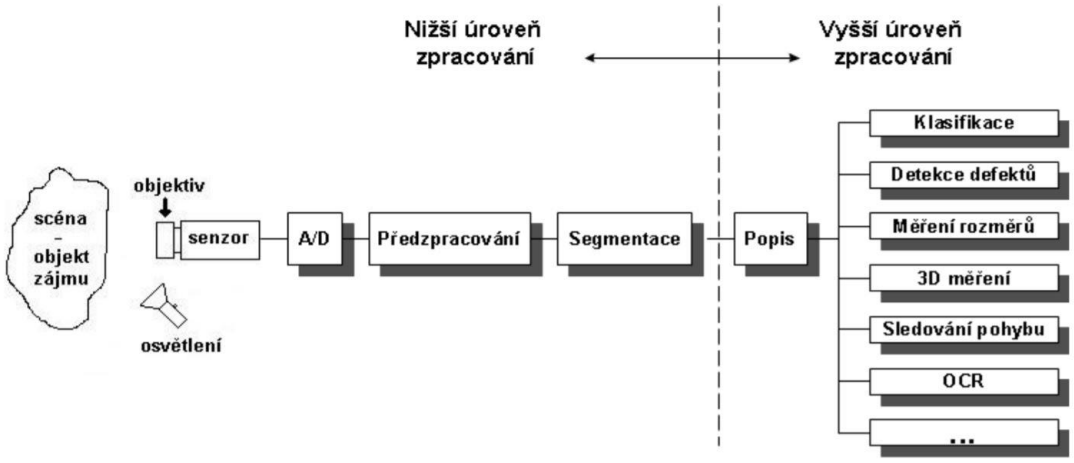
Program pro první kategorii provede detekci vyfotografovaného papíru, následně opraví perspektivu a další geometrické deformace, odstraní přebytečné stíny a na závěr převede obrázek na černobílý, resp. šedotónový s možností volby úrovně šedi. Volba úrovně šedi má smysl právě při tisku upraveného obrázku. Uživatel si navolí počet barev – jasových úrovní v obrázku tak, aby kvalita obrázku byla pořád dostačující, a zároveň se tímto krokem ušetří toner. Obdobně se budou upravovat snímky z druhé kategorie s tím, že v tomto případě proběhne detekce mřížky, ve které se budou nacházet žádané informace.

Obsah práce je rozdělen na čtyři hlavní části. První část má za úkol vysvětlit pojem digitální obraz a řetězec jeho zpracování. V další části je provedena analýza vlastností zdrojových snímků a jejich vliv na tvorbu a funkčnost programu. Ve třetí části proběhne detailní rozbor jednotlivých metod počítačového vidění vhodných pro tuto úlohu. Zároveň bude ukázáno využití těchto metod při tvorbě programu. Na závěr proběhne zhodnocení úspěšnosti programů, jejich nedostatků a možných zlepšení.

1 ŘETĚZEC ZPRACOVÁNÍ OBRAZU

V následujících podkapitolách budou popsány jednotlivé kroky vedoucí ke zpracování obrazu od pořízení snímku až po výsledný zpracovaný, tedy upravený obrázek.

Předtím než začneme využívat metody počítačového vidění pro úpravu snímků, je potřebné uvědomit si, co představuje digitální obraz a jak s takovým obrazem pracuje počítač. Při pořizování snímku se obecně převádí analogový (spojitý) obraz na digitální (diskrétní). Z matematického pohledu takový obraz tvoří matice o N řádcích a M sloupcích. Digitální obraz je tedy mapa bodů v rovině, přičemž jeden takový bod nazýváme pixelem. Pixel je nejmenší jednotka v obraze a značí jasovou hodnotu v daném místě roviny. Převod spojitého obrazu na digitální a jeho zpracování je možné popsat řetězcem zpracování obrazu, který se rozděluje na 5 základních částí.



Obrázek 1-1: Jednotlivé úrovně zpracování obrazu

1.1 Snímání

Snímání obrazu je prvním krokem v řetězci zpracování obrazu. Jedná se o převod optické veličiny na elektrický signál, který je spojitý v čase i úrovni. Za vstupní složku se považují jednotlivé světelné paprsky dopadající na detektor - obrazový snímač. Na tomto místě se zachytávají informace o pořizované fotografii jako je intenzita světla a barva. Tento proces si tedy lze představit jako prostorovou digitalizaci a platí, že čím je plocha snímače větší, tím kvalitnější bude výsledný snímek. Pro výrobu snímačů se využívají především technologie CCD a CMOS. Signál získaný ze snímání je nakonec potřebné digitalizovat, o čemž pojednává následující kapitola. Ve fázi snímání je zároveň

důležité, pokud je to možné, co nejvíce eliminovat nežádoucí vlivy jako je špatné osvětlení, nevhodné stíny nebo fotografování ze špatného úhlu. V opačném případě můžou tyto faktory zkomplikovat funkčnost a spolehlivost programu.

1.2 Digitalizace

Po provedení snímání obrazu následuje jeho digitalizace - převod spojitého signálu na diskrétní. Pro tento účel se využívají A/D převodníky, které dokážou vzorkovat a kvantovat vícerozměrný signál. Vzorkování signálu, tedy diskretizaci v časové oblasti, si je možné představit jako postupné odebírání jednotlivých vzorků hodnot ze signálu s určitou vzorkovací frekvencí. Na druhou stranu kvantování signálu je digitalizace v amplitudové oblasti, kdy dochází k diskretizaci oboru hodnot signálu. Oba procesy vedou ke ztrátě informací v signálu, a proto je důležité pracovat se správným rozměrem detektoru snímání. Při příliš malém rozměru se v obrazu ztrácí důležité informace. Při velmi velkých rozměrech zase neúměrně stoupají nároky na výpočetní výkon. Těmito problémy a jejich vlivy na tvorbu programu se budeme zabývat v dalších kapitolách této práce.

1.3 Předzpracování

Předzpracováním se obraz připravuje tak, aby byly odstraněny přebytečné informace, resp. se zvýraznily žádané rysy v obrazu, to znamená vyhledání vhodných metod počítačového vidění pro správné řešení úkolů této práce. Tyto metody jsou detailně popsány v kapitole č. 3.

1.4 Segmentace obrazu

Segmentace obrazu je v naší aplikaci nejtěžší krok při zpracovávání obrazu. Cílem je oddělení části obrazu tak, abychom dostali oddělenou část obrazu, se kterou pak chceme dále pracovat. Výsledek segmentace je velmi závislý na struktuře obrazu a množství šumu. Časté je překrývání objektů nebo nerovnoměrné světelné podmínky v obrazu. Pro náš program to bude znamenat správnou detekci objektů - papíru, resp. mřížky tak, aby bylo možné detekovaný objekt oddělit od nepotřebného pozadí.

1.4.1 Druhy segmentace

Segmentace se rozděluje na několik druhů:

- **Prahování** - Nejstarší a nejjednodušší metoda segmentování. Základem je rozdílnost intenzity jasu u každého pixelu. Po určení prahu se všechny hodnoty

jasu pod touto úrovní považují za pozadí. Všechny hodnoty nad prahem se považují za objekty. Průchod obrazem stačí udělat jenom jednou.

- **Segmentace založená na detekci hran** – Hledáme oblasti obrazu, ve kterých dochází k výrazné změně jasu. Častým problémem je však příliš široké pásmo, kde se jas mění nebo zašumění obrazu. Je vhodné předem znát informace o objektech, jako jsou rozměry a barva.
- **Segmentace založená na hledání oblastí** – Často využívaná pro zašumělé obrazy, protože na ně funguje lépe než jiné metody. Obraz je dělený na menší části a zjišťuje se, jestli dané části splňují homogenitu. Kritériem homogenity může být například jas, pokud víme, že objekty jsou tmavé a pozadí světlé.
- **Segmentace založená na porovnání se vzorem** – Porovnává se známý objekt s obrazem, ve kterém se daný objekt hledá. Nepřesnosti jsou způsobené převážně šumem a zkreslením.

1.5 Popis objektů

Po úspěšné segmentaci následuje popis objektů. Je to poslední část v řetězci zpracovávání obrazu a patří do vyšší úrovně zpracování. Cílem je porozumět dané části obrazu pro jeho jednodušší zpracování. Pro tento program se bude jednat o správnou korekci geometrických deformací.

2 ZDROJOVÉ SNÍMKY A JEJICH VLASTNOSTI

Tato část obsahuje popis vlastností zdrojových snímků a poukazuje na problémy, které můžou tyto vlastnosti způsobovat při vytváření programu.

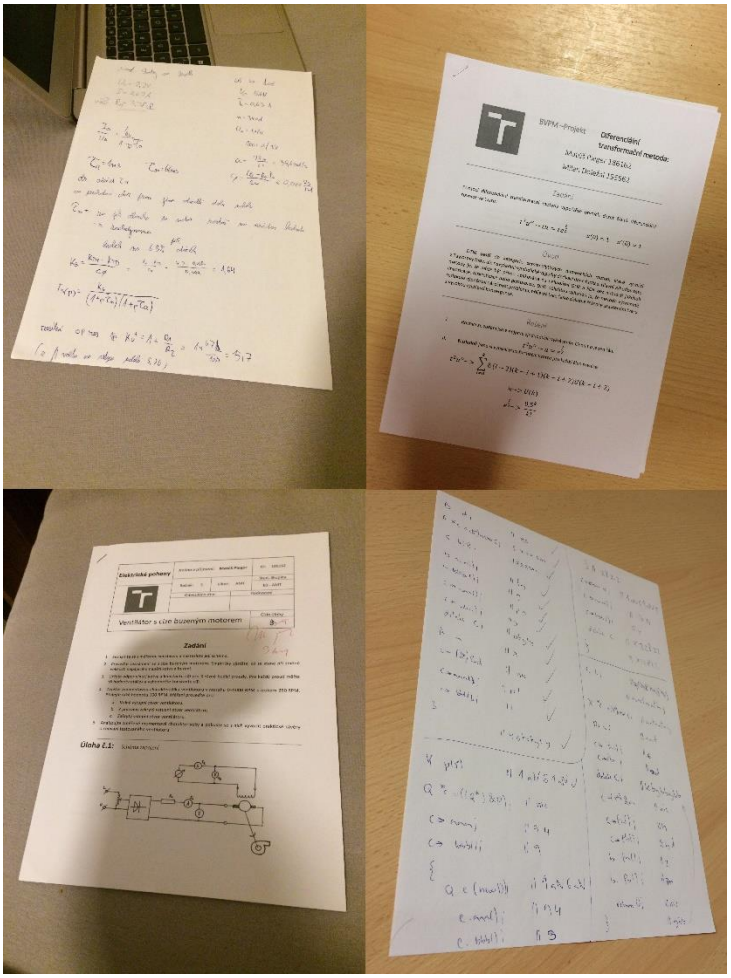
Jak už bylo načrtnuto v úvodu, úkolem této práce je spolehlivá detekce žádaných objektů, úprava geometrických deformací a stínů a nakonec převod snímků na černobílé, resp. šedo-tónové s možností volby úrovně šedi. Proto bylo potřeba vytvořit dostatečně rozsáhlý data-set vstupních snímků. Pro snímky byla dána jedna podmínka, a to že snímky musí být vyhotovené tak, aby se v obrazu nacházel celý papírový dokument, resp. čtyřúhelníková mřížka. Pro korekci geometrických deformací je totiž důležitá detekce hledaného objektu v obrazu a ta se bude dít především pomocí detekce hran. Proto je potřebné, aby byly viditelné všechny obrysy zmiňovaného objektu. Data-set byl vytvořen tak, aby obsahoval fotografie vyfotografované z různých úhlů a za rozmanitých světelných podmínek. Je rozdělený na dvě sady a snímky jsou obvykle ve vysokém rozlišení pro dobré zachování detailů i po úpravě snímků. Většina byla vyfotografována mobilním telefonem Apple iPhone 5S.

První sada pojmenovaná „paper_documents“ obsahuje 30 snímků papírových dokumentů a to jak dokumentů tištěných, tak i dokumentů, kde je jejich obsah psaný ručně. Snímky jsou zhotovené tak, aby obsahovaly celý papír včetně rohů. Počet snímků byl zvolen s důrazem na to, aby bylo možné dostatečně přesně vyhodnotit funkčnost programu.

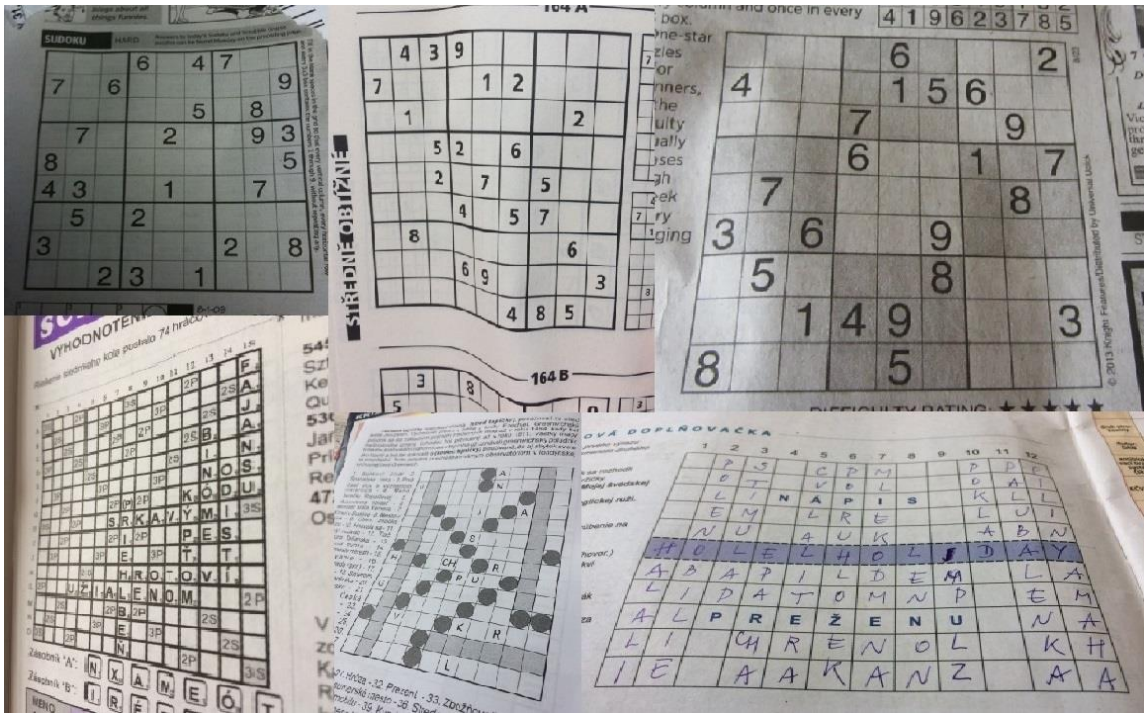
Druhá sada s názvem „sudoku_crosswords“ obsahuje 30 snímků sudoku a 20 křížovek nebo jiných hlavolamů s čtyřúhelníkovým ohraničením vytištěných na papír.

Pro zajímavost obsahuje data-set několik běžných obrázků z různých oblastí života, jsou to např. fotografie přírody, různých předmětů v domácnosti atd. Na těchto obrázcích je možné vyzkoušet operaci redukce bitové hloubky a následně pozorovat výsledky.

Protože má program fungovat co nejspolehlivěji, je nevyhnutelné pochopit faktory, které ovlivňují úspěšnost detekce a úprav snímků. Rozbor nežádoucích vlivů na úspešnost programu je uveden v následujících podkapitolách.



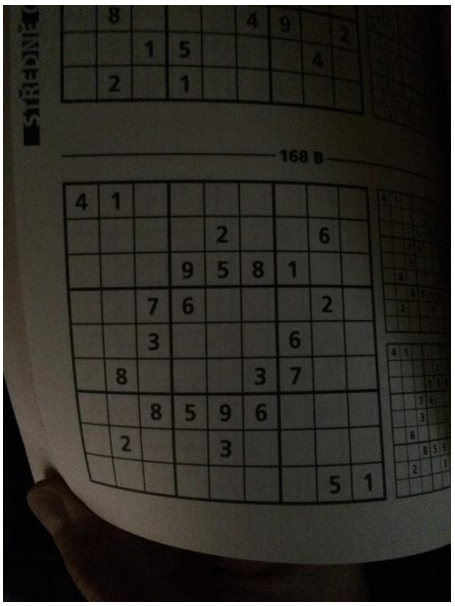
Obrázek 2-1: Příklady vstupních snímků z kategorie papírové dokumenty



Obrázek 2-2: Příklady vstupních snímků z kategorie sudoku a křížovky

2.1.1 Špatné osvětlení a přebytečné stíny

Velmi častý problém při fotografování představuje špatné osvětlení a tvorba stínů. Problémem je jak nízká míra osvětlení, tak příliš velká intenzita světla. Obě mají za následek ztrátu detailů v obrazu.



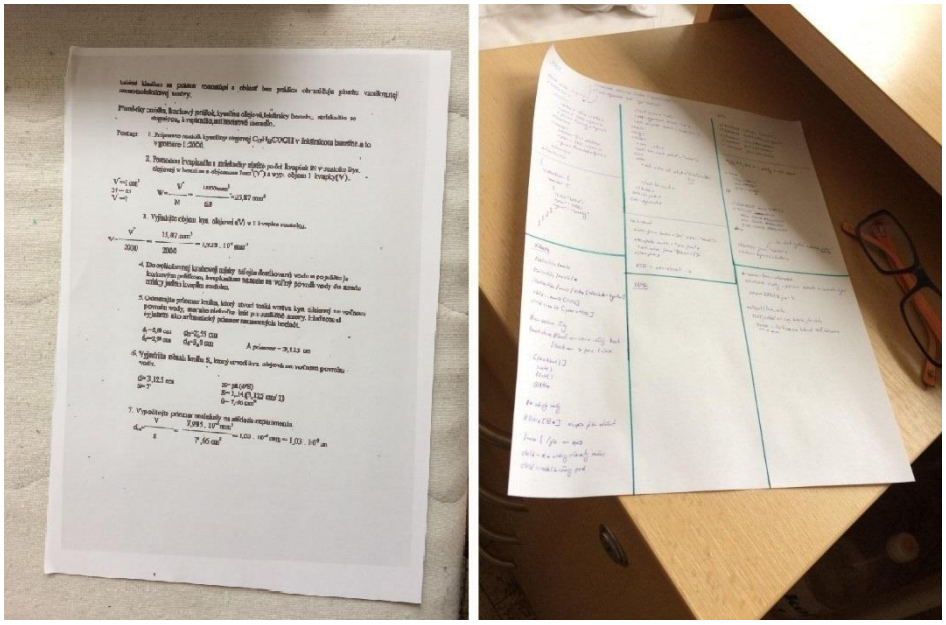
Obrázek 2-3: Příklad vstupního snímku se značným stínem - špatným osvětlením ve velké části obrazu

Jak můžeme vidět na obrázku 2-3 nízká míra osvětlení – stín v jedné části obrazu má za následek částečné splynutí hledaných hran objektu s pozadím. To může zkomplikovat detekci nebo ji úplně znemožnit. To samé platí pro vstupní snímky, kde na hrany hledaného objektu svítí příliš intenzivní světlo

Pokud se ale detekce povede, je dál nutné najít takové metody počítačového vidění, které potlačí stíny a současně zanechají potřebné informace v obrazu. O odstraňování stínů pojednává kapitola 3.8.

2.1.2 Problematické pozadí

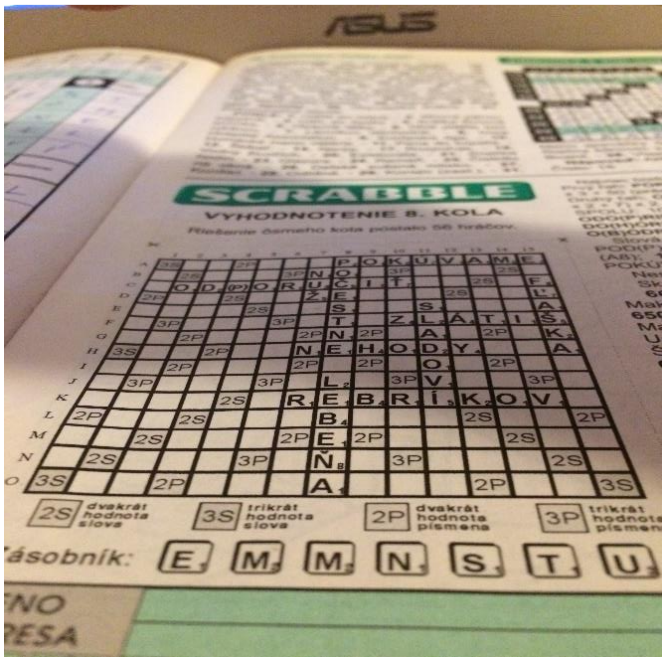
O úspešnosti detekce a následném zpracování rozhoduje i pozadí, které se kolem hledaného objektu nachází. Pokud je pozadí příliš světlé a tedy splývá s barvou papíru, je pravděpodobné, že program nebude umět objekt detekovat. Příklad takového vstupního snímku je na obrázku 2-4 vlevo. Napravo je zase příklad obrázku, kde pozadí obsahuje mnoho ostrých hran a špatné osvětlení v pravé horní části papíru.



Obrázek 2-4: Příklady problematického pozadí

2.1.3 Chyba perspektivy

Chyba perspektivy vzniká při fotografování z nevhodných úhlů. U takových snímků se pak ztrácí důležité detaily především v té části obrazu, která byla nejdál od čočky fotoaparátu. Dá se říct, že naprostá většina snímků obsahuje menší nebo větší chybu perspektivy. Tato chyba je ale poměrně snadno řešitelná aplikováním vzorců pro výpočet projektivní transformace. Geometrické deformace jsou rozebírány v kapitole 3.7.



Obrázek 2-5: Obrázek vyfotografovaný se špatnou perspektivou

3 METODY POČÍTAČOVÉHO VIDĚNÍ A REALIZACE PROGRAMU

V této sekci bude popsán výběr jednotlivých metod vhodných pro řešení detekce a úpravy žádaných částí vstupních snímků a realizace programu

Obor počítačového vidění poskytuje velké množství různých metod pro zpracování vstupních snímků. Pro správnou aplikaci těchto metod je proto potřebná komplexní znalost problému a možností řešení. Mezi základní problémy patří velké množství dat, což vede k vysokým nárokům na počítačovou paměť. Další problém představuje velká variabilita obrazových dat. Z toho důvodu se algoritmy pro zpracování obrazů obvykle specializují na konkrétní úlohy. V opačném případě je potřebné programovat algoritmy tak, aby byly dostatečně robustní vůči vstupním snímkům různé kvality, světelných podmínek atd. To na druhou stranu vede k jiným problémům jako je například zpomalení programu nebo velká chybovost programu. Z některých snímků i přes veškerou snahu není možné po jejich zpracování extrahovat požadovanou informaci nebo dosáhnout chtěného výsledku z důvodu velkého zašumění obrazu a obecně nedostatečné kvality.

Programy pro řešení úloh této práce budou naprogramovány v jazyce Python s využitím knihovny OpenCV. Tato svobodná knihovna obsahuje mnoho užitečných funkcí pro práci s obrazem. Výhodou těchto funkcí je kromě užitečnosti v programech i značná rychlost provedení.

Cíle programů jsou následující:

1. Předzpracování obrazu tak, aby byla snadná detekce hran, resp. rohů hledaného objektu.
2. Oprava geometrických zkreslení nalezeného objektu.
3. Odstranění nežádoucích stínů tak, aby výsledný obrázek obsahoval jenom podstatné informace na bílém pozadí či papíře.
4. Převod upraveného obrázku na černobílý, resp. šedotónový s umožněním výběru úrovně šedi.

Provedení jednotlivých programů bude rozděleno do dvou kategorií zmíněných v kapitole Úvod. V případě neuspokojivého výsledku upraveného výstupního obrázku, program navíc nabídne uživateli možnost manuální detekce hledaného objektu použitím kurzoru a označením jednotlivých rohů.

3.1 Změna rozměrů – softwarové vzorkování

Jak už bylo popsáno v kapitole 1.2, každý nasnímaný obraz prochází diskretizací, tedy vzorkováním. Prvotní vzorkování probíhá na samotném snímači fotoaparátu a označuje se jako hardwarové vzorkování. Buňky snímače tvoří obvykle ortogonální mřížku a k vzorkování dochází automaticky při snímání na samotný maticový snímač.

Snímky z fotoaparátu mají ale obvykle rozlišení, které je příliš velké a při úkolech zpracování obrazů neúměrně zatěžují výpočetní výkon počítače. Je proto vhodné rozměry vstupních snímků zmenšit tak, aby nedocházelo k významné ztrátě detailů.

3.1.1 Geometrická transformace - měřítko

K samotné změně rozměrů se používá geometrická transformace typu měřítko. Obecně se jedná o transformaci souřadnic bodů ze vstupního obrazu na výstupní obraz pomoci transformační matice.

Mezi základní metody pro změnu rozměru obrazu patří:

- **Metoda nejbližšího souseda** - Tato metoda spočívá v tom, že se v obrazu zopakují jednotlivé řádky či sloupce nebo se řádky či sloupce vynechají - v závislosti na tom, zda se provádí převzorkování do vyššího rozlišení obrazu nebo do nižšího rozlišení obrazu. Jde o nejjednodušší a nejrychlejší metodu. Tato metoda zachovává ostrost vodorovných a svislých hran obrazu, ale hrany a ostré linie jsou zubaté. V obraze se tedy objevují jasné „kostičky“.
- **Bilineární interpolace** - Uvažuje 2x2 okolí kolem nového počítaného bodu. Počítaní hodnoty nového bodu se děje na základě lineární interpolace po sloupcích a po řádcích. Tyto hodnoty se následně sčítají. Výsledný obraz po použití této metody je mnohem jemnější, může být ale jemně rozostřený.

Obrázek 3-1: Ukázka bilineární interpolace

- **Bikubická interpolace** - Metoda, která dosahuje ještě lepších výsledků než bilineární interpolace. Uvažuje 4x4 okolí a výpočet hodnoty nového pixelu je dán aproximací stávajících pixelů polynomickou funkcí třetího stupně.

Protože se jedná o zmenšování snímků, na tom, kterou metodu použijeme, až tak nezáleží. Všechny metody po zmenšení obrazu totiž produkují uspokojivé výsledky. Každopádně zvolíme metodu bikubické interpolace z důvodu, že poskytuje nejpřesnější výsledky a zároveň je pořád dostatečně rychlá.

Abychom měli všechny snímky konstantně velké alespoň v jednom rozměru, je nutné zvolit koeficient zmenšení. Během testování programu bylo zjištěno, že při šířce obrázků o hodnotě 400 je program pořád schopen fungovat spolehlivě. Zvolíme tedy koeficient rovný velikosti šířky původního obrázku podělen hodnotou 400. Následně využijeme funkci *resize*, která jako vstupní parametry požaduje novou šířku a výšku obrázku. Posledním parametrem je interpolační metoda.

Většina fotografií v data-setu je o rozměru 2448x3264. Velikost takové fotografie je po komprimaci přibližně 711 kB. Po provedení zmenšení obrázku se jeho velikost sníží na 100 kB. Jedná se tedy o značnou úsporu paměti a výpočetního výkonu počítače. Protože ale takové zmenšení může způsobit např. nečitelnost textu v dokumentu, využívá se toto zmenšení jen pro zrychlení detekce hledané oblasti. Pro další úpravy se využívá obrázků v původní velikosti.

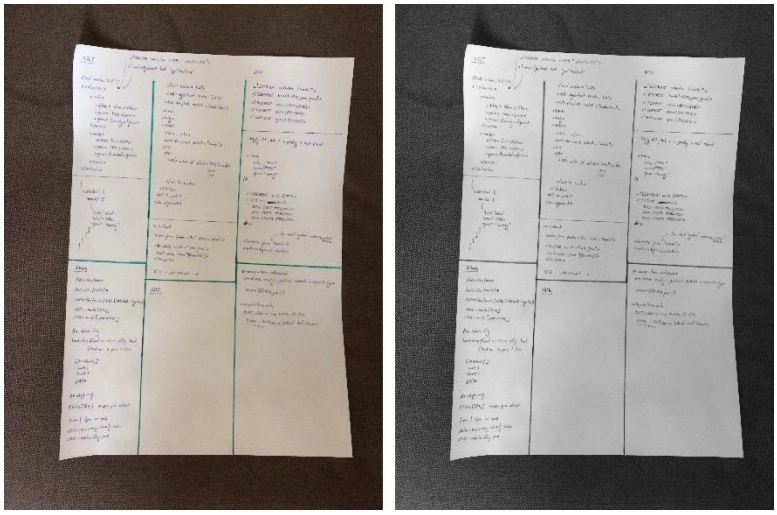
3.2 Převod barevných snímků na šedotónové

Mnoho metod pro zpracování obrazu vyžaduje vstupní šedotónové (grayscale) snímky. Vstupní snímky z fotoaparátu jsou ale obvykle barevné. Pokud uvažujeme nad obvyklým barevným modelem RGB, tak takový obrázek je složen ze tří dvojrozměrných matic. Každá matice definuje jednu z barevných složek (červená, zelená a modrá). Matice určují intenzity jednotlivých spektrálních složek a teprve jejich aditivním složením vznikne barevný obraz. Tento proces si je možné představit jako překrytí tří různých obrazů a následného součtu jednotlivých prvků. Pro převod barevného obrázku na šedotónový se dá použít prostý aritmetický průměr všech tří složek. Lepších výsledků se ale dosáhne vážením každé složky empiricky zjištěným koeficientem a následným sčítáním složek (1). Jednotlivé koeficienty se liší z důvodu rozdílné citlivosti lidského oka na jednotlivé spektrální složky.

$$G(x,y) = 0.299 \cdot Red(x,y) + 0.587 \cdot Green(x,y) + 0.114 \cdot Blue(x,y) \qquad (1)$$

Je vhodné uvědomit si, že šedotónový obrázek se nerovná černobílému obrázku. Zatímco černobílý obrázek obsahuje jenom dvě jasové úrovně, šedotónový jich obsahuje víc. Obvykle je to 256 úrovní neboli 2^8. Toto číslo je označované jako bitová hloubka obrazu a představuje celkové množství možných jasových úrovní v obrazu. Pojem bitová hloubka souvisí s kvantizováním obrazu a je podrobněji vysvětlen v kapitole 3.10.

V programu se převod vstupního snímku na šedotónový děje aplikováním zmíněného vzorce přes funkci *cvtColor*. Jelikož je barevný obrázek složený ze tří matic, je jeho velikost při stejné bitové hloubce 3x větší než velikost šedotónového obrázku. Tímto krokem tedy ušetříme další paměť a velikost typického obrázku z data-setu se zmenší na 33 kB. Výsledek převodu barevného obrázku na šedotónový je možné vidět níže.



Obrázek 3-2: Ukázka převodu barevného obrázku na šedotónový

3.3 Potlačení šumu

Po zmenšení a převodu vstupního snímku na šedotónový přistoupíme k jeho předzpracování pro zabezpečení co nejlepších podmínek detekce hran. V tomto momentě se algoritmus dělí na dvě větve. Snímky z kategorie č. 2, sudoku a křížovky, jsou připravené pro detekci a další úprava snímků nevede k lepším výsledkům detekce hran. U kategorie č. 1 je potřebné snímky dále předzpracovat.

Prvním krokem je odstranění šumu. Šum v obrazu jsou data bez významu, která vznikla jako nechtěný vedlejší produkt jiných aktivit, především při digitalizaci a přenosu obrazu. Generovaný elektrický signál je mnohokrát ovlivněn jiným elektromagnetickým zářením, teplotními kmity krystalové mřížky, teplotou polovodičových součástek a integrovaných obvodů atd. Pro větší úspěšnost detekce hran v obrazu je vhodné se přebytečného šumu zbavit.

Šum se rozděluje na několik typů:

- **„Pepř a sůl“** - Zrnění, impulsní šum - u obrazů s více jasovými úrovněmi
- **Gaussovský šum** - Hustota pravděpodobnosti šumu má Gaussovo rozložení
- **Poissonovský šum** - u senzorů pracujících jako čítač fotonů (CCD)
- **Bílý šum** - Idealizovaný šum, používá se pro simulace nejhorších degradací obrazu, ve výkonovém spektru má rovnoměrně zastoupeny všechny frekvence

- **Kvantizační šum** - Vzniká, když není použit dostatečný počet jasových úrovní
- **Aditivní šum** - Vzniká při přenosu obrazu nebo snímání

Pro potlačení šumu se nabízí různé druhy lokálních lineárních i nelineárních filtrů, které se aplikují na vstupní obraz různými způsoby.

3.3.1 Průměrování

Průměrování je nejjednodušší metoda pro potlačení šumu. Tato metoda počítá hodnotu pixelu jako průměr z jeho okolí. Matematicky se jedná o operaci konvoluce částí obrazu s aplikovaným konvolučním filtrem typu dolní propust. Konvoluci si lze představit jako postupné prostorové posouvání převráceného filtru po obrazu a následné stanovení odezvy pro každý bod. To znamená, že každý bod ve zpracovávaném okolí obrazu je vážen příslušným koeficientem filtru a následně jsou všechny body sečteny. Výsledkem průměrování je potlačení vysokých frekvencí v obrazu. Nevýhodou je rozmazávání ostrých hran. Z tohoto důvodu je tento druh potlačování šumu nevhodný pro úlohy využívající detekci hran a v této práci ho tudíž nepoužijeme.

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Obrázek 3-3: Příklady konvolučních filtrů typu dolní propust pro metodu průměrování

3.3.2 Hledání mediánu, modusu, maxima a minima

Tyto filtry pracují na principu hledání požadovaných typů hodnot v malém okolí, jsou tedy nelineární. Výhodou je, že nerozmazávají hrany, nevýhodou je porušovaní tenkých čar, což se nehodí především při hledání uzavřených kontur po detekci hran. Mediánový filtr je ale v programu použit při procesu odstraňování stínů.

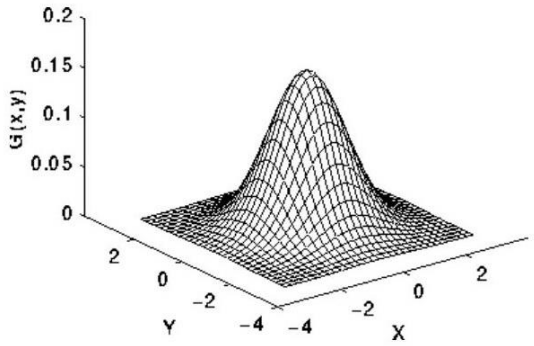
3.3.3 Filtr s Gaussovým rozložením

Pro odstranění přebytečného šumu ve vstupním obrazu a pro zlepšení detekce hran se pro naši úlohu nejvíce hodí filtr s Gaussovym rozložením. Ten je koncepčně podobný filtru průměrování, jelikož také používá operaci konvoluce. Pro vytvoření konvolučního filtru ale využívá vzorce pro 2D distribuce Gaussova rozložení v podobě rozptylové funkce.

Gaussův filtr neporušuje tenké čáry a zároveň dokáže lépe zachovat hrany jako prostý filtr průměrování.

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1



Obrázek 3-4: Konvoluční filtr a Gaussovo rozložení v 2D

3.4 Morfologické operace

Morfologické operace jsou založené na vlastnostech bodových množin. Jsou realizovány jako relace mezi zpracovávaným obrazem (diskrétní obraz je množina bodů) a strukturním elementem, který také představuje množinu bodů. Strukturní element může být různý, vždy ale musí mít označený počátek. Morfologie se využívá především pro zjednodušení struktury objektu nebo naopak pro její zvýraznění. Jelikož je potřebné pro kategorii č. 1 správně detekovat hledaný objekt, morfologické operace nám poslouží pro odstranění nepotřebných hran a zvýraznění těch hledaných.

Mezi základní operace patří dilatace a eroze. Kombinací těchto dvou operací je možné dosáhnout morfologického otevření a uzavření. Provádí se obvykle na černobílých - binárních obrázcích. Tyto operace je ale možné rozšířit i na šedotónové nebo barevné obrázky, což je případ této práce.

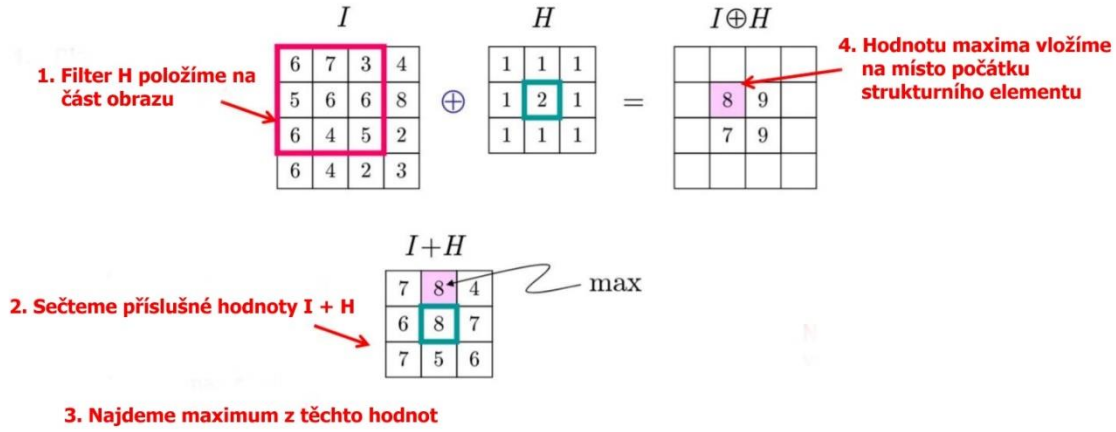
3.4.1 Dilatace

Dilatace je morfologická operace vektorového součtu prováděná mezi obrazem a strukturálním elementem. Výsledkem je zaplnění děr a jiných malých mezer v obrazu, což je výhodné pro zlepšení detekce hran.

U binárních obrázků operace probíhá tak, že se strukturní element pohybuje po obrazu a porovnává hodnotu počátku s aktuální hodnotou pixelu v obrazu. Pokud se shodují, všechny pixely okolí, které kopírují svou pozicí pixely se stejnou hodnotou v strukturním elementu jako hodnota počátku, se přepíšou ve výstupním obrázku na hodnotu počátku.

V programu se ale využívá dilatace šedotónového obrázku. V tomto případě se nejprve odpovídající si jasové hodnoty strukturního elementu a části obrazu sečtou a následně se najde maximum z těchto hodnot. Nalezená hodnota maxima se zapíše

na danou pozici ve výstupním obrázku. Názorný příklad dilatace šedotónového obrázku se strukturním elementem se zvýrazněným středem vidíme na obrázku 3-5.



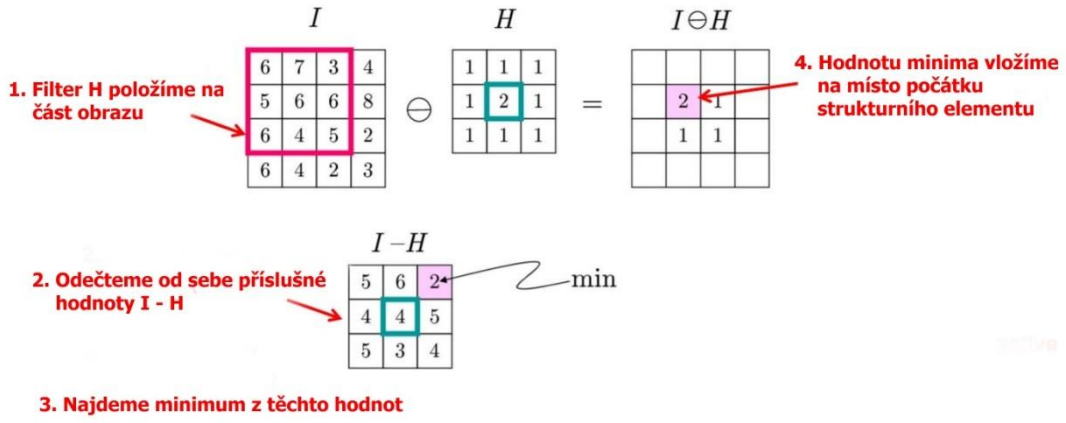
Obrázek 3-5: Příklad dilatace šedotónového obrázku

3.4.2 Eroze

Druhou základní morfologickou operací je eroze. Používá se pro zjednodušení struktury objektů. Je to duální operace k dilataci. Objekty zjednodušuje tak, že velmi malé objekty zmizí a složité objekty spojené tenkými čárami rozloží na několik jednodušších objektů. Matematicky se jedná o operaci vektorového rozdílu.

V binární rovině strukturní element znovu „hledá“ pixely stejné s hodnotou počátku. Pokud se takový pixel najde, porovnávají se pixely okolí. Pokud jsou všechny pixely okolí totožné s pixely elementu, do výstupního obrázku se zapíše hodnota počátku. V opačném případě se na výstup zapíše opačná hodnota.

U šedotónových obrázků se opakuje stejný postup jako u dilatace, místo maxima se ale hledá minimum.



Obrázek 3-6: Příklad eroze šedotónového obrázku

3.4.3 Morfologické otevření

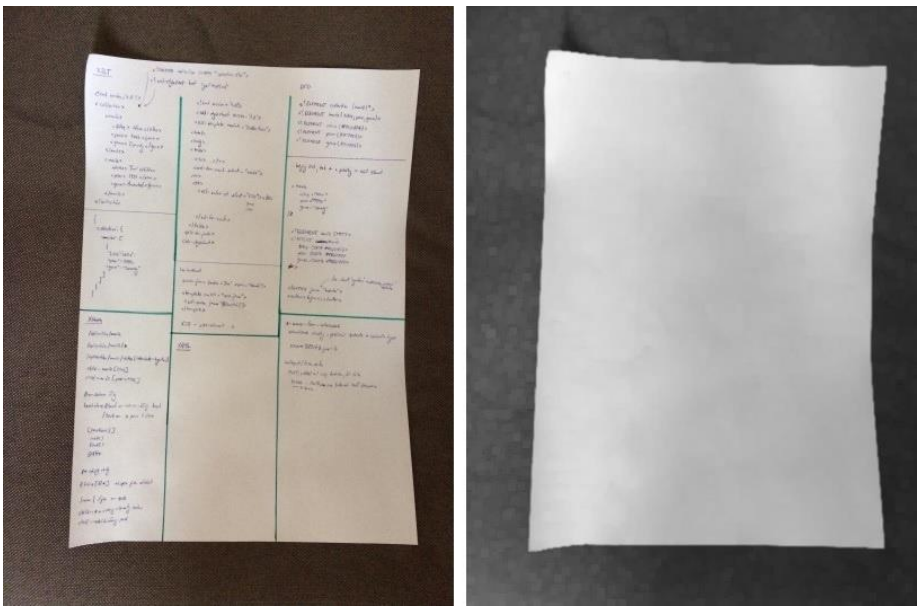
Morfologické otevření je operace, která kombinuje operaci eroze následovanou operací dilatace. Způsobuje oddělení objektů spojených úzkou šíjí a tím zjednodušuje strukturu objektů.

3.4.4 Morfologické uzavření

Operace morfologického uzavření naopak využívá dilataci následovanou erozí. Tato operace spojí objekty, které jsou blízko u sebe, zaplní díry a vyhladí obrys. Pro účely tohoto programu se využívá u první kategorie vstupních snímků (papírové dokumenty). Uzavření zjednoduší strukturu snímků, odstraní nepotřebný text a zároveň zachová celistvost hledaných okrajů papíru.

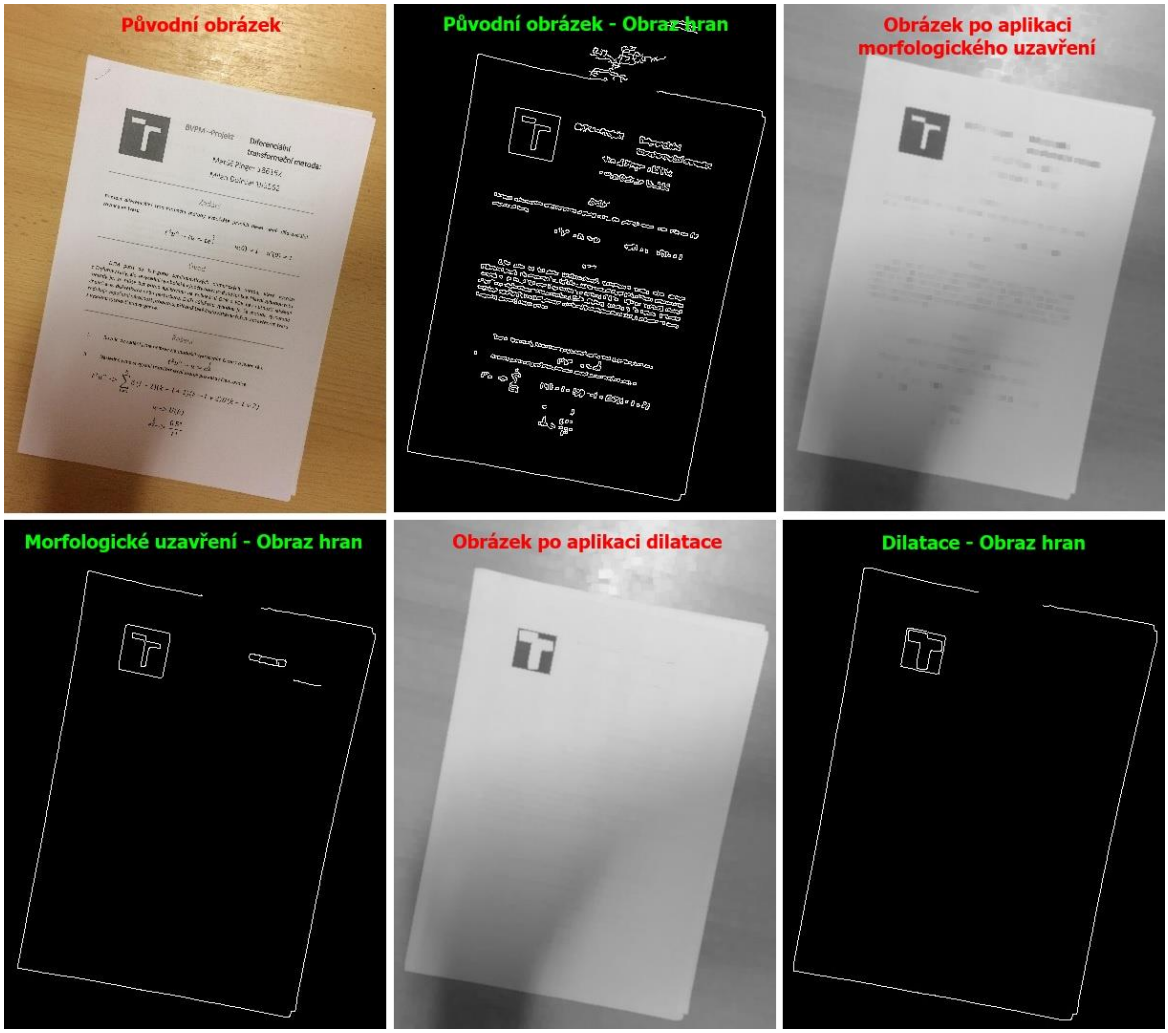
Pro náš případ je nejlepší použití morfologického uzavření. Tím dosáhneme spojení drobných mezer, které by nám mohly překážet při detekci hran a zároveň zjednodušíme strukturu objektu. Samotná detekce hran je pak podrobně popsána v kapitole 3.5. Jako strukturní element při operaci morfologického uzavření využijeme čtvercovou matici o rozměru 5x5 naplněnou jedničkami. Rozměr 5x5 byl zvolen proto, že při testování větších rozměrů bylo zjištěno, že se kvalita detekce postupně zhoršuje.

Po aplikaci na vstupní snímky nám pořád v obrazu můžou zůstat hrany po některých malých objektech. Proto na zpracovávaný obrázek aplikujeme dodatečnou dilataci. Na obrázku 3-7 vidíme, že použitím morfologických operací dokážeme vyhladit obraz tak, že v něm zůstanou a zvýrazní se jen ty pro nás zajímavé hrany papíru.



Obrázek 3-7: Aplikace morfologických operací na vstupní snímek z kategorie paper_documents

Při testování detekce hran (kapitola 3.5) pak vidíme rozdíl v detekci před aplikováním morfologického uzavření, po aplikaci a nakonec ještě s dodatečnou dilatací.



Obrázek 3-8: Srovnání výsledků detekce hran po aplikaci vybraných metod předzpracování obrazu

3.5 Detekce hran

Detekce hran se používá pro nalezení míst v obrazu, které nám poskytují informace o tvaru a velikosti hledaných objektů. Hrany můžou být výsledkem změny jasu, barvy, stínů nebo textury ve vstupním snímku. Jedná se o místo v obrazu s velkou změnou jasové funkce. Změna může být jak pozvolná, tak skoková. V ideálním případě bychom po aplikování detektoru hran měli dostat spojité čáry ohraničující objekty v obrazu. Obraz však obsahuje mnoho nepotřebných informací jako je třeba šum. Proto bylo potřebné, tak jak bylo popsáno v předešlých kategoriích, vstupní snímky předzpracovat.

Detekci hran je možné rozdělit na dva typy, a to na základě stupně derivace obrazu, kterou používají.

3.5.1 Detekce hran užitím první derivace obrazu

Metody, které používají první derivaci, jsou založeny na výpočtu gradientů ve zpracovávaném obrazu. Gradient (2) popisuje změnu jasových úrovní obrazové funkce a zapisujeme ho ve tvaru:

$$\nabla f(x,y) = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} \tag{2}$$

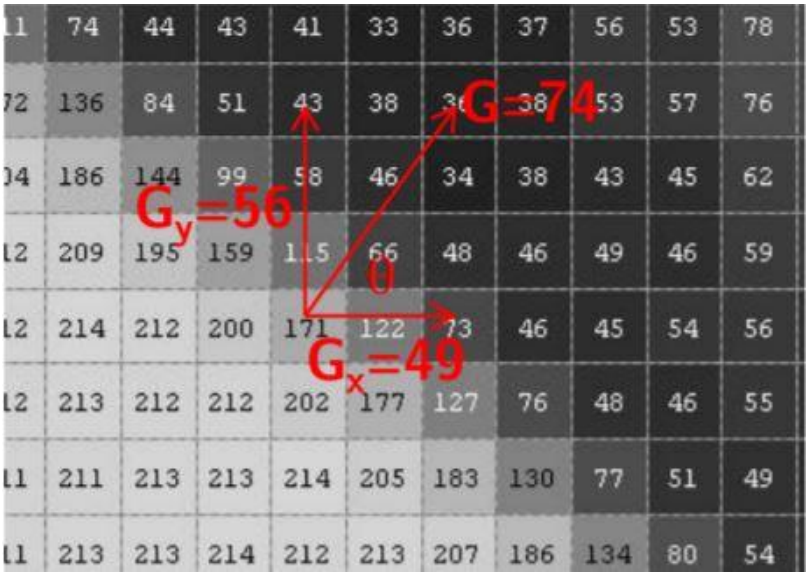
Výpočet gradientů spočívá v aplikaci operace konvoluce vstupního obrazu a hranového operátoru. Hranový operátor si lze představit jako matici obvykle o rozměru 3x3. Samotný gradient je vektor. Jelikož je změna obrazové funkce dvojrozměrná, obsahuje složky x a y. Každá ze složek ale udává změnu obrazové funkce jenom podél jedné z os. Proto je potřebné vypočítat gradienty v obou složkách. To provedeme využitím hranového operátoru pro daný směr a následnou konvolucí. Celkovou velikost gradientu pro dané místo pak vypočítáme na základě vzorce:

$$G[f(x,y)] = \sqrt{{G_x}^2 + {G_y}^2} \tag{3}$$

Pro výpočet směru gradientu se používá vzorec:

$$\theta(x,y) = \arg(G_{x,}G_y) \tag{4}$$

Při práci s diskrétním obrazem se jednotlivé složky gradientu navíc aproximují diferencemi. Analogicky se vypočítají velikosti a směry gradientů.



Obrázek 3-9: Gradient obrazové funkce a výpočet jeho velikosti

Mezi operátory založenými na výpočtu gradientů nalezneme několik druhů. Jedná se o operátory:

- **Robertsův kříž** - Jedná se o výpočetně nenáročný operátor, kvůli malému počtu zpracovávaných pixelů je ale hodně citlivý na šum.
- **Prewittové operátor** - Obsahuje symetrický operátor o obvyklém rozměru 3x3 a je považován za jeden ze základních operátorů pro detekci hran. Je definován ve čtyřech variantách otočení. Výsledná odezva se redukuje buď na jeden směr natočení, nebo se kombinuje více variant do jednoho výsledku.
- **Sobelův operátor** - Podobný Prewittové operátoru, rozdíl je pouze ve váze koeficientů u sousedů ve čtyř-okolí vzhledem ke směru detekce.
- **Kirschův operátor** - Nesymetrický operátor o rozměru 3x3, také je definován ve čtyřech variantách otočení.
- **Robinsonův operátor** - Stejný mechanismus jako předešlé operátory.
- **Cannyho operátor** - Založený na Sobelově operátoru. Využívá potlačení pixelů, které nejsou považované za část hran a hysterezi se spodním a horním prahem.

Práci s operátory si lze představit podobně jako při potlačování šumu využitím matematické operace konvoluce vstupního obrazu s daným operátorem. Důležitým aspektem těchto operátorů je, že reagují jenom v jednom zvoleném směru, a proto je obvykle potřeba počítat jednotlivé odezvy s různě otočenými maskami. Pro získání celkové odezvy je nutné odezvy jednotlivých směrů sčítat.

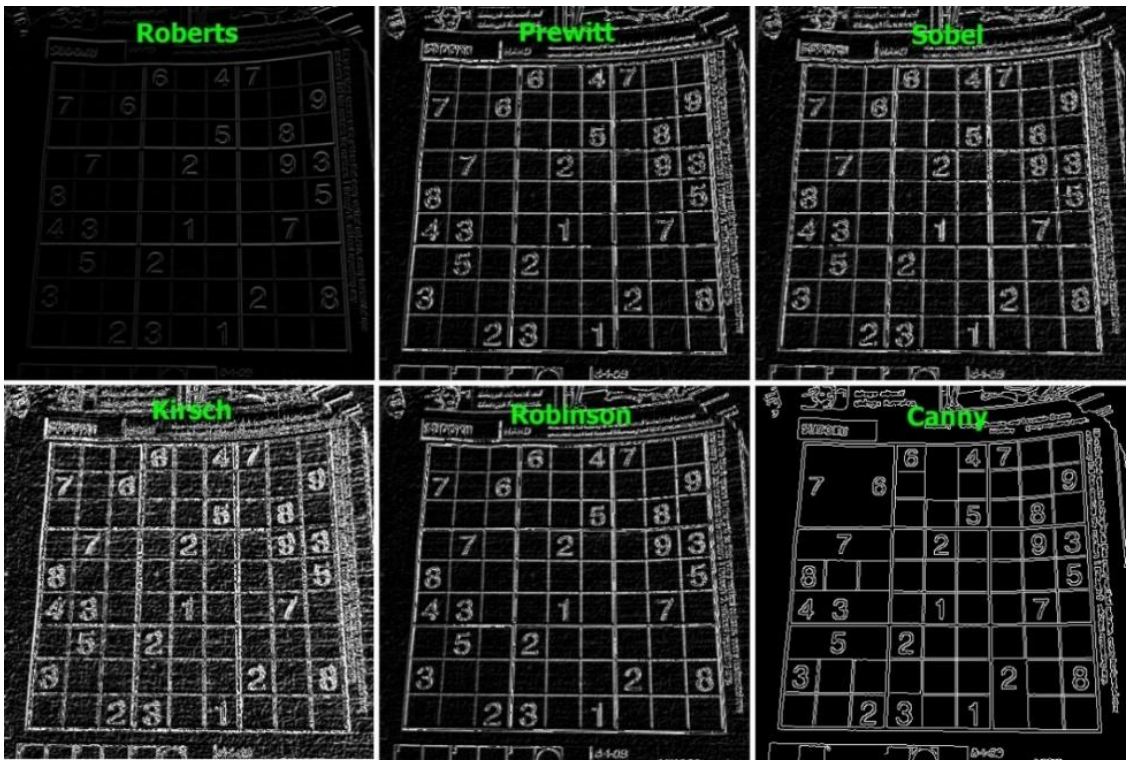
Protože existuje několik operátorů, vyvstává otázka, který použít pro řešení naší úlohy. Nejlepším způsobem, jak to zjistit, je jednotlivé operátory vyzkoušet. V programu můžeme využít předpřipravené operátory z knihovny OpenCV nebo si definovat matice operátorů a provést operaci konvoluce.

$$R=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \quad P=\begin{bmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{bmatrix} \quad S=\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix} \quad Ro=\begin{bmatrix}1 & 1 & -1\\ 1 & -2 & -1\\ 1 & 1 & -1\end{bmatrix}$$

$$K=\begin{bmatrix}3 & 3 & -5\\ 3 & 0 & -5\\ 3 & 3 & -5\end{bmatrix} \quad L_4=\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix} \quad L_8=\begin{bmatrix}1 & 1 & 1\\ 1 & -8 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Obrázek 3-10: Přehled některých základních hranových operátorů (Robertsův, Prewittové, Sobelův, Robinsonův, Kirschův, Laplacián v 4-okolí, Laplacián v 8-okolí)

Po aplikaci operátorů na jeden ze vstupních snímků jsme získali následující výsledky:

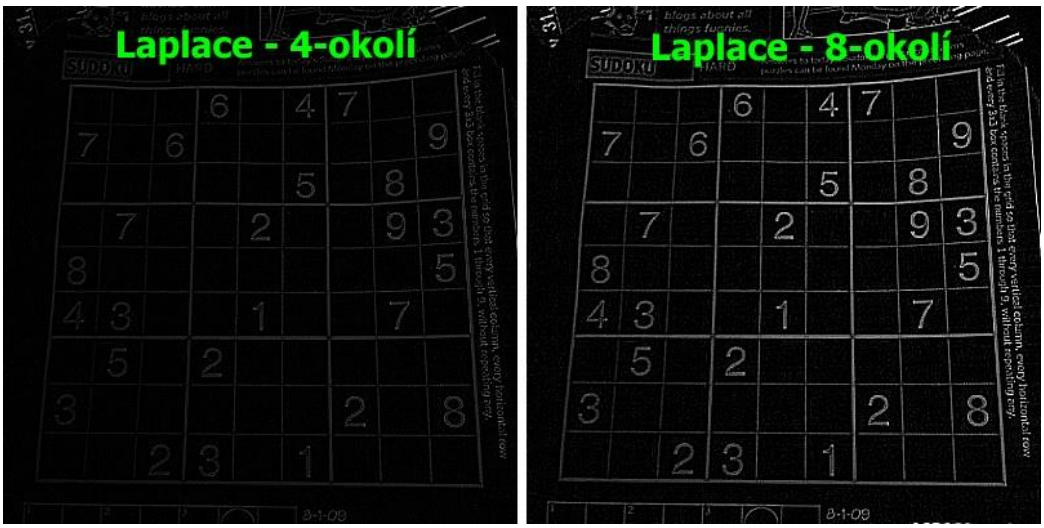


Obrázek 3-11: Detekce hran po aplikaci vybraných hranových operátorů

Pro správnou detekci objektů je nejpodstatnější spolehlivá detekce jejich okrajů. V tomto ohledu nám nejlepší výsledky poskytuje Cannyho detektor hran. Tento detektor je jeden z nejvyužívanějších detektorů v oboru zpracování obrazu. Jak je popsáno výše, využívá Sobelův operátor a navíc ponechává jenom pixely v okolí s největší změnou jasové funkce. Tento proces se nazývá ztenčování hran. Nakonec tento operátor využívá hysterezi, kde je uživatelem nastaven spodní a horní práh. Všechny pixely pod prahem jsou potlačeny. Pixely nad prahem jsou automaticky akceptovány jako hrany. Pixely v intervalu jsou ponechány, jestliže jsou spojeny s pixely s hodnotami nad prahem. V samotném programu se nejvíc osvědčily hodnoty intervalu mezi 40 a 200.

3.5.2 Detekce hran užitím druhé derivace obrazu

Detektory hran, které využívají druhou derivaci obrazu jsou početně menší skupinou a patří zde především Laplaceův operátor ve čtyř-okolí a osmi-okolí. Tyto operátory využívají koncept použití druhé derivace obrazu, na základě které je možné hledat inflexní body hran, tedy průchody nulou druhé derivace. Tento způsob je výpočetně jednodušší než hledání extrémů první derivace, výsledek je zároveň přesnější. Další výhodou je, že operátory využívající druhou derivaci obrazu jsou invariantní vůči rotaci. To znamená, že není potřebné počítat odezvy pro všechny směry otočení.



Obrázek 3-12: Detekce hran na základě operátorů využívajících druhou derivaci

Laplaceovy operátory nám pro vstupní snímky poskytují celkem dobrou detekci okrajů, a to především laplacián v 8-okolí. Problém ale nastává u míst s nižší hodnotou jasu, kde tyto operátory kreslí hrany s nedostatečným kontrastem. Proto v tomto programu dále použijeme pouze Cannyho detektor hran.

3.5.3 Detekce hran pomocí adaptivního prahování obrazu

Jiným způsobem, jak docílit detekci hran v obrazu, je využitím adaptivního prahování. Ve zkratce se jedná o úpravu jasových hodnot v obrazu na základě určeného prahu v malém okolí obrazu. Jelikož při detekci hran převádíme šedotónové obrázky na černobílé, bude to znamenat, že všechny hodnoty nad určeným prahem převedeme na hodnotu 255, bílou barvu. Všechny hodnoty pod prahem budou mít jasovou úroveň rovnou 0. Tím docílíme vyobrazení hran objektu na černém pozadí. Tento způsob detekce hran je spíše doplňkový, v některých případech může ale poskytovat lepší výsledky detekce hran. Jedná se zejména o kategorii obrázků sudoku, kde pomoci adaptivního prahování dokážeme zaplnit mezery u velmi tenkých hran a tím pádem využít detekci objektů pomoci hledání kontur, jak bude vysvětleno v kapitole 3.6.1.

Prahování obrazu a výpočet lokálních prahů využijeme při odstraňování stínů a bude podrobně rozebráno v kapitole 3.9.2 a 3.10.2.

3.6 Detekce objektů

Poté, co se nám podařilo detekovat hrany vstupních snímků, můžeme přistoupit k detekci pro nás zajímavých objektů. Pro kategorii č. 1 se bude jednat o detekci vyfotografovaných papírových dokumentů, u kategorie č. 2 budeme hledat mřížku vyfotografovaných obrázků sudoku, křížovek, tabulek atd. Je nutné si uvědomit, že nás

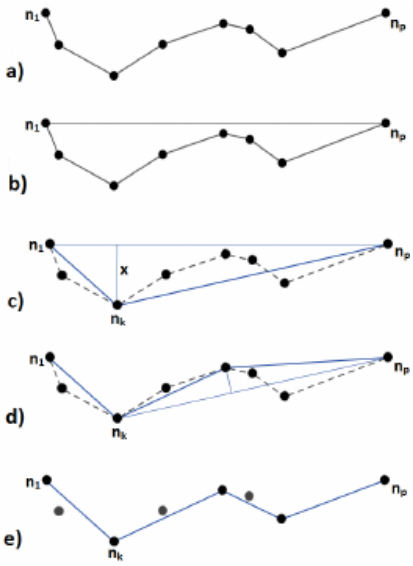
zajímají především souřadnice rohů hledaných objektů. Tyto souřadnice totiž později využijeme pro opravu jejich perspektivy. Pro detekci objektů a tedy rohů se nabízí dvě možnosti řešení, a to detekce hledáním kontur objektů a detekce hledáním samotných rohů.

3.6.1 Detekce objektů pomocí hledání kontur

Detekované hrany ve vstupních předzpracovaných snímcích se hodí pro hledání kontur všech objektů v obrazu. Kontury jsou spojité čáry o stejné jasové úrovni ohraničující všechny objekty různých tvarů nacházející se v obrazu. Při detekci hran je ale celkem častým problémem nespojitá detekce. Může se stát, že hranový detektor detekuje většinu hran, ale vlivem špatného osvětlení nedetekuje část hrany nacházející se právě v místě s problematickým osvětlením. Proto se tento způsob detekce objektů hodí pouze u snímků, které netrpí problémy, které jsme si vysvětlovali v kapitole č. 2. Z tohoto důvodu bude detekce objektů v programu rozdělena tak, že se program nejprve pokusí o detekci hledáním kontur, a pokud se tato detekce nepovede, program vyzkouší druhý způsob detekce samotným hledáním rohů hledaného objektu.

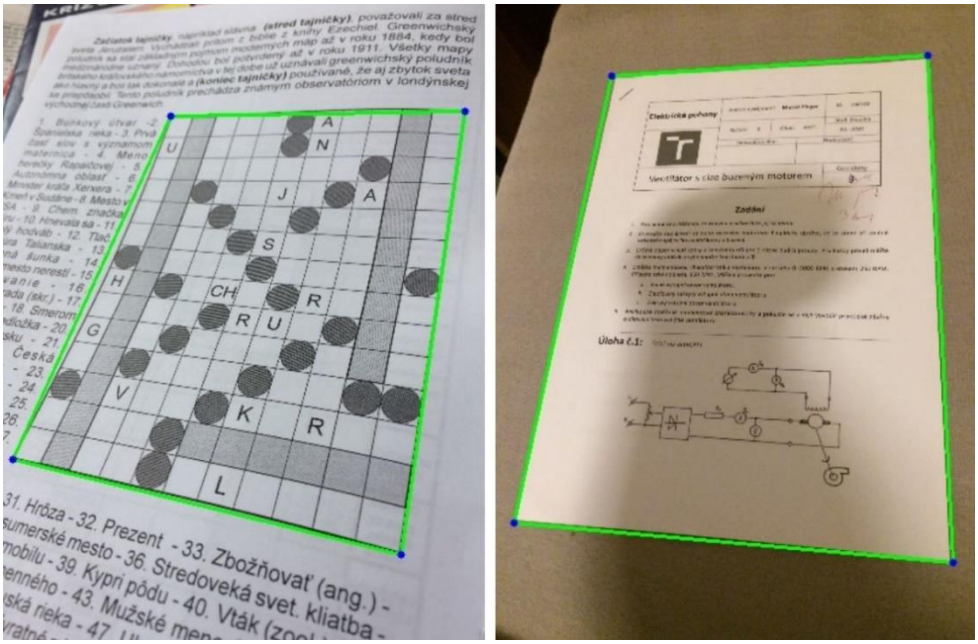
Hledání kontur je možné provést využitím knihovní funkce s názvem *findContours*. Tato funkce vrátí pomocí pole souřadnice jednotlivých bodů sestavující nalezenou konturu. Jednotlivých kontur, ale může být v obrazu relativně hodně. Pro ušetření paměti je výhodné jednotlivé kontury seřadit podle obsahu, který ohraničují a vybrat si jenom 5 největších. K tomu nám pomůže funkce *contourArea*, pomocí které vypočítáme obsahy nalezených kontur.

Po provedení popisovaných kroků přejdeme k analýze nalezených kontur. Jelikož budeme hledat čtyřúhelníkové objekty (papírové dokumenty, mřížky, tabulky atd.), musíme umět určit, které z nalezených kontur jsou, nebo se podobají čtyřúhelníkovým útvarům. Obvykle není hledaný objekt perfektním čtyřúhelníkem. Pro zjednodušení nalezených útvarů využijeme funkci *approxPolyDP*, kterou chceme využít tak, aby aproximovala konturu na geometrický útvar se čtyřmi vrcholy. Samotná funkce využívá Ramer-Douglas-Peuckerova algoritmu, který dekomponuje křivku složenou z mnoha úseků čar na podobnou křivku s menším počtem vrcholů. Jako argumenty funkce přijímá pole souřadnic bodů tvořící nalezenou konturu a parametr přesnosti. Zároveň vrátí hodnoty souřadnic vrcholů, rohů aproximované křivky. Parametr přesnosti nám vyjadřuje maximální vzdálenost, kterou může aproximovaná křivka dosáhnout vůči původní křivce. Tento parametr je poměrně těžké určit tak, aby vyhovoval všem zpracovávaným snímkům a zároveň se zpracovávaná kontura upravila na útvar se čtyřmi vrcholy. Obecně se ale číslo přesnosti pohybuje v mezích od 1 do 10 % velikosti obvodu kontury.



Obrázek 3-13: Ukázka využití Ramer-Douglas-Peuckerova algoritmu pro zjednodušení křivky

Z toho důvodu je nejlepší vyzkoušet více hodnot přesnosti a následně, pokud má aproximovaná uzavřená křivka čtyři vrcholy, zjistit obsah, který tento útvar pokrývá. Předpokládáme, že pokud je obsah větší, než jedna třetina obsahu celého snímku, povedlo se nám nalézt hledaný objekt.



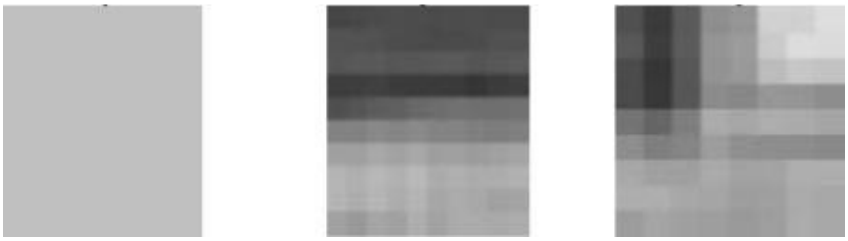
Obrázek 3-14: Detekce objektů z obou kategorií pomocí hledání kontur

3.6.2 Detekce objektu pomocí hledání rohů

Při zpracovávání vstupních snímků nastávají případy, kdy ani po jejich předzpracování není možné detekovat kontury hledaných objektů, které jsou v celém

rozsahu spojité. V takových situacích se nabízí řešení přímého hledání rohů v nespojitých konturách.

Rohy jsou místa v obrazu, která vykazují vysokou změnu jasové funkce ve více než jednom směru. Jedná se o místa, která jsou co nejméně podobná svému okolí. Jak můžeme vidět na obrázku 3-15, vlevo se nachází typ lokální oblasti v obrazu, která nevykazuje žádnou změnu jasové funkce. Ve středu pozorujeme lokální oblast vykazující změnu jasové funkce ve vertikálním směru, jedná se tedy o přítomnost hrany. Vpravo se nachází typ lokální oblasti se změnou jasové funkce v obou směrech, což indikuje přítomnost rohu na daném místě.



Obrázek 3-15: Porovnání různých typů lokálních míst v obrazu

Pro hledání rohů existuje několik algoritmů. Jedním z nejpoužívanějších a nejrychlejších je Harrisův detektor rohů. Funkci tohoto detektoru si vysvětlíme na rovnici (5):

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \quad (5)$$

Jak vidíme, detektor počítá rozdíly hodnot mezi původním a posunutým obrázkem. Proměnné *u* a *v* představují posunutí obrazu o dané hodnoty. Ve vzorci se počítají rozdíly v jasových intenzitách mezi původním bodem a posunutým bodem, přičemž funkce *w* představuje okénkovou funkci. Obvykle se jedná o čtverec s hodnotami 1 v místě, které analyzujeme a 0 na ostatních místech obrazu. To zaručuje spočítání rozdílů jasových hodnot a následnou sumaci jenom pro určité okolí.

Princip Harrisova detektoru tedy spočívá v tom, že pokud budou rozdíly hodnot jasových úrovní mezi původním a posunutým obrázkem malé, resp. blízké nule, o roh se jednat nebude. Pokud naopak budou rozdíly velké, je velká šance, že je dané okolí rohem. Abychom mohli s jistotou určit, které části obrázku obsahují rohy, upravíme vzorec pomoci Taylorovy řady a dostaneme:

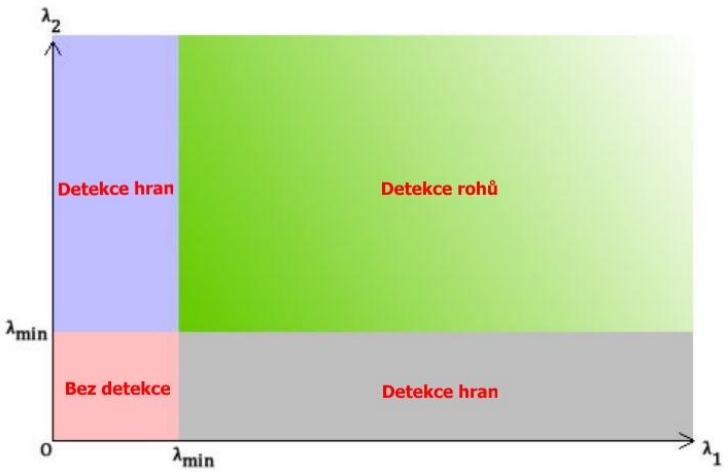
$$E(u,v) \approx [u\ v] * M * \begin{bmatrix} u \\ v \end{bmatrix} \quad (6), \text{ kde M představuje: } M = \sum w(x,y) \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{bmatrix} \quad (7)$$

Teorie Harrisova detektoru nám pak dále napovídá, že k určení, jestliže se v dané části obrazu roh opravdu nachází, potřebujeme vypočítat hodnotu parametru *R*. Tento parametr vypočítáme následovně:

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{8}$$

Hodnoty obou proměnných λ z rovnice (8) vypočítáme tak, že od matice *M* odečteme jednotkovou matici vynásobenou proměnnou λ a následně položíme determinant výslední matice rovný nule. Pro parametr *R* platí, že pokud toto číslo nabývá vysokých hodnot, analyzované okolí v obrazu je rohem.

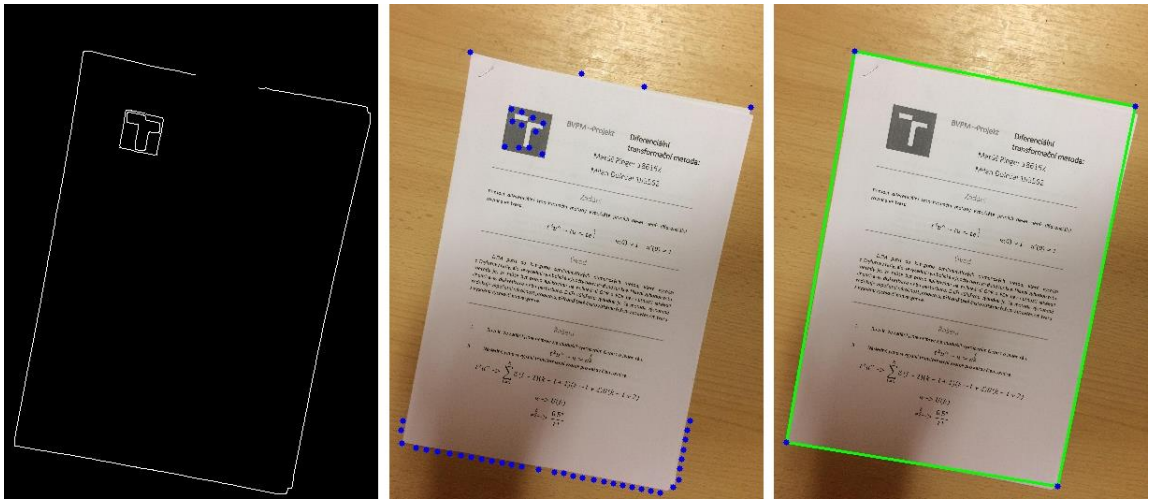
Harrisův detektor poskytuje poměrně dobré výsledky při hledání rohů. Naše zdrojové snímky, ale často obsahují špatné světelné podmínky, nebo jsou fotografované ze značných úhlů – chyba perspektivy. Pro správnou detekci hledaného objektu je ale nutné, aby detektor dokázal najít všechny relevantní rohy. Z toho důvodu je vhodné využít Shi-Tomasiho detektor rohů. Tento detektor je téměř shodný s Harrisovym detektorem, pro výpočet parametru *R*, ale používá jiný vzorec. Myšlenka tohoto detektoru spočívá v tom, že hodnota *R* je vyčíslená na základě hodnot obou proměnných λ. Pokud jsou obě hodnoty λ nad stanoveným minimem, sledované okolí je rohem. Tento detektor je robustnější a poskytuje lepší výsledky než Harrisův detektor.



Obrázek 3-16: Detekce rohů Shi-Tomasiho detektorem

V našem programu využijeme knihovní funkci *goodFeaturesToTrack*. Funkce pracuje se šedotónovým snímkem a nabízí možnost výběru, kolik rohů se detekuje. Vždy platí, že podle toho jaký počet vybereme, dostaneme jenom daný počet nejvýraznějších rohů v obrazu. Je proto důležité vybrat takovou hodnotu, abychom i v případě, že se některý z hledaných rohů papírového dokumentu nachází v místě obrazu, kde je špatně rozeznatelný, dokázali tento roh detekovat. Pro potřeby programu se osvědčila hodnota

50. Poté, co získáme souřadnice jednotlivých rohů, musíme určit, které rohy patří samotnému analyzovanému objektu. Vypomůžeme si předpokladem, že se rohy dokumentu budou nejspíš nacházet nejblíž k rohům celkového obrazu. Pro levý horní roh dokumentu tedy musí platit, že součet souřadnic musí být co nejmenší. Obdobně pro levý dolní roh bude platit, že nejmenší musí být rozdíl souřadnic. Naopak, pro pravý dolní roh bude platit, že součet souřadnic bude největší a pro pravý horní roh budeme hledat maximum rozdílu souřadnic. Tímto způsobem dostaneme z pole souřadnic mnoha rohů právě čtyři rohy charakterizující polohu hledaného objektu.



Obrázek 3-17: Použití detekce rohů po nespojité detekci kontur

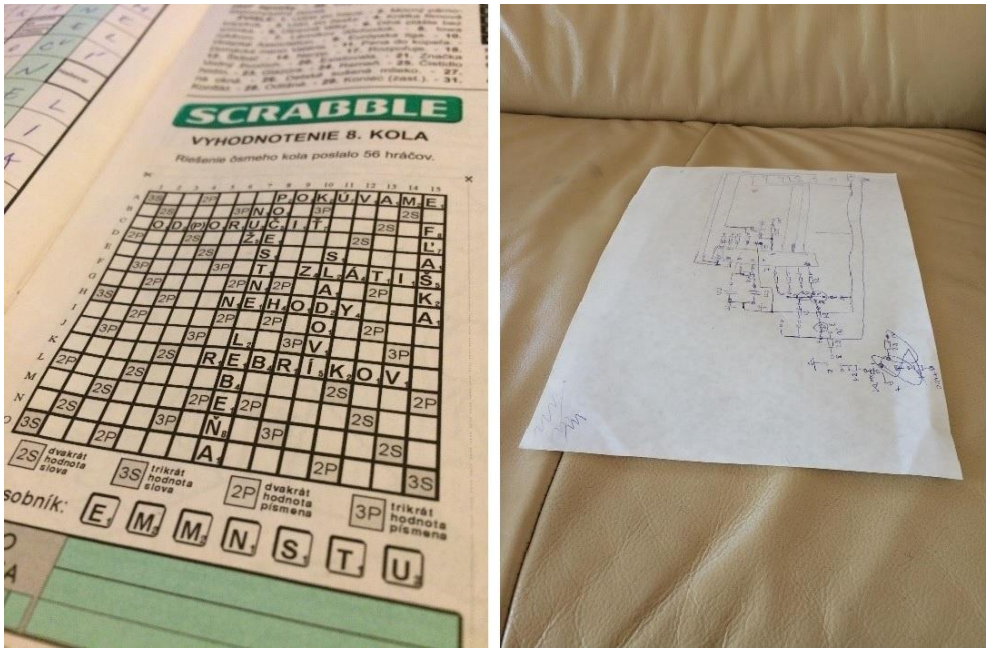
Je potřeba přiznat, že tento způsob detekce nemusí být vždy úspěšný. U vstupních snímků obsahujících nevhodné hrany v pozadí, může nastat situace, že využitím předpokladu o umístnění rohů co nejblíže ke krajům samotného obrazu, program vybere roh nacházející se v okolí nějaké přebytečné hrany. To se může stát z důvodu, že taková hrana bude zasahovat až na samý roh celého obrazu a tím pádem takový roh bude splňovat podmínky matematického předpokladu lépe, než skutečný roh hledaného objektu. Tento způsob detekce je proto spíše doplňkový, v případě, že program nedokáže detekovat spojitou konturu ohraničující pozici hledaného objektu.

Pokud selžou oba druhy detekce, poslední možností je nechat detekci rohů na uživateli programu. Ten má proto po provedení detekce, jestliže není s výsledkem spokojen, možnost vybrat si manuální detekci rohů. Ta funguje tak, že uživatel pomoci kursoru vybere čtyři rohy hledaného objektu a program pak dále pracuje s těmito souřadnicemi.

3.7 Oprava perspektivy

Po úspěšné detekci hledaných objektů, přistoupíme k opravě perspektivy vstupních snímků. Perspektiva je optický jev, který způsobuje to, že se vzdálené objekty jeví zdánlivě menší, než objekty blízké. Chyba perspektivy se vyskytuje u fotografii, jestliže rovina snímače fotoaparátu není rovnoběžná s liniemi, které mají být ve fotografii paralelní. To v našem případě způsobí, že jednotlivé hrany hledaných objektů nemusí být rovnoběžné s hranami protilehlými a subjektivně se nám bude objekt jevit, jako kdyby se oddaloval od fotoaparátu. Z takového pohledu se ale obsah jednotlivých papírových dokumentů čte celkem špatně. Části, které se nachází nejdále od fotoaparátu a jeví se, jako menší, mohou být celkem nečitelné. Z tohoto důvodu je vhodné perspektivu snímků opravit tak, aby byl úhel pohledu na vyfotografované dokumenty kolmý na rovinu, na které se nacházejí.

Pro opravu perspektivy a obecně pro korekci všech druhů zkreslení se využívají geometrické transformace. Geometrickou transformaci si lze představit, jako namapování jednotlivých souřadnic obrazových bodů na nová místa v souřadnicovém systému ve výstupním obrazu. To se děje podle zvolené transformační matice. Při takové transformaci, ale vzniká problém, kde celočíselným souřadnicím ze vstupního obrazu mohou po transformaci odpovídat neceločíselné souřadnice. V tomto případě nelze určit přesně jasovou hodnotu v daném místě obrazu a je nutné tuto hodnotu najít buď pomoci zaokrouhlení, což není moc přesné nebo použitím interpolace jasových hodnot. Jednotlivé typy geometrických transformací a určení, která je nejvhodnější pro náš případ, rozebereme v následujících kapitolách.



Obrázek 3-18: Vstupní snímky z obou kategorii se značnou chybou perspektivy

3.7.1 Afinní transformace

Do první kategorie geometrických transformací patří afinní transformace. Je to typ transformací, který zachovává kolinearitu a poměr vzdáleností mezi body. To znamená, že všechny body v obrazu ležící na jedné přímce, zůstanou i po transformaci na té samé přímce a zároveň se vzdálenost mezi středem této přímky a krajem nezmění. Afinní transformace se dají vyjádřit jako:

$$\begin{aligned} x' &= a_0 + a_1 x + a_2 y \\ y' &= b_0 + b_1 x + b_2 y \end{aligned} \tag{9}$$

Tyto rovnice i přes jednoduchost matematického vyjádření, obsahují čtyři základní geometrické transformace. Obecné vyjádření výsledných souřadnic transformovaných bodů pak vypadá následovně:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_0 \\ b_1 & b_2 & b_0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{10}$$

Mezi základní druhy afinních transformací patří:

- **Posunutí** – Operace posunutí (11) realizuje posunutí vstupního obrazu o hodnotu posunu ve směru x, čemu v rovnici odpovídá koeficient a_0 a ve směru y, což značí koeficient b_0. Tato transformace, ale pro nás nemá v programu využití.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a_0 \\ 0 & 1 & b_0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{11}$$

- **Měřítko** – Využitím transformace typu měřítko (12) jsme schopní zvětšit, resp. zmenšit obraz. Tuto operaci jsme aplikovali při zmenšování vstupních snímků, viz kapitola 3.1. Hodnoty a_1 a b_2 nám specifikují koeficienty zvětšení nebo zmenšení podél os x a y.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{12}$$

- **Rotace** – Pomoci rotace (13) můžeme otočit obrázek o libovolný úhel kolem počátku souřadnicového systému. Tato operace využívá čtyři koeficienty {a_1, a_2, b_1, b_2} z rovnic ve formě goniometrických funkcí sinus a kosinus, kde argumentem je samotná hodnota úhlu natočení. Tuto transformaci bychom mohli využít

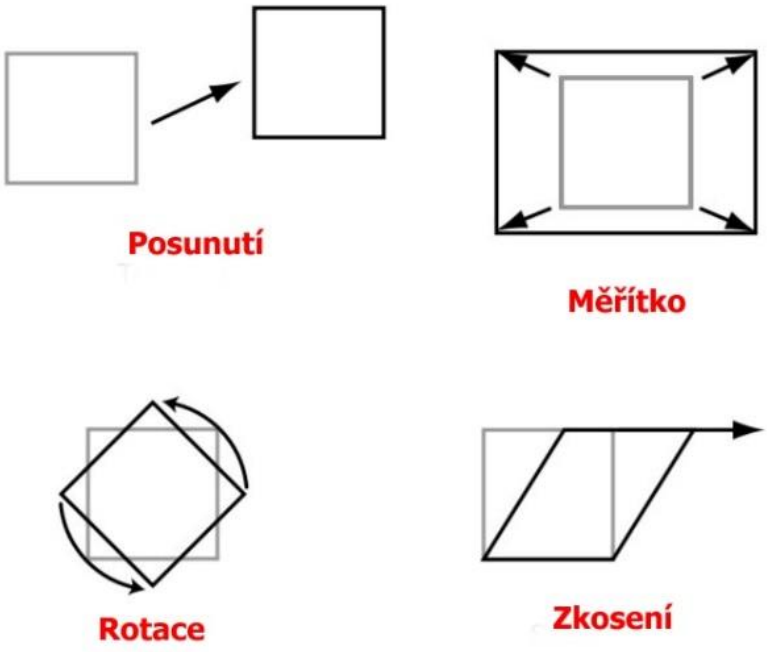
pro opravu horizontu vstupních snímků, nedokážeme ale pomoci ni korigovat chybu perspektivy.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{13}$$

- **Zkosení** – Zkosení (14) využívá členy a_2 a b_1 a natáčí objekty buď ve směru osy x, nebo osy y podle zvolených koeficientů. Ani v tomto případě, ale nedokážeme vyřešit náš problém s chybou perspektivy.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & a_2 & 0 \\ b_1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{14}$$

Jak vidíme, pomoci afinních operací je možné objekty posouvat nebo natáčet. Dá se tedy říct, že pomoci afinních transformací dokažeme detekovaným objektem pohybovat ve směru osy x, y nebo současně oběma směry. To ale neřeší náš problém, kde vstupní snímky potřebujeme opravit tak, abychom na ně měli pohled rovnoběžný s rovinou, na které detekované objekty leží. Pro tento případ využijeme projektivní transformace, které rozšiřují možnosti afinních transformací a umožňují pohybovat objektem ve směru osy z.



Obrázek 3-19: Grafické znázornění základních typů afinních transformací

3.7.2 Projektivní transformace

Jak už bylo naznačeno, projektivní transformace rozšiřuje možnosti afinních transformaci. Pomoci této transformace dokážeme perspektivu opravit tak, aby byl výsledný pohled kolmý na rovinu objektu. Obecné vyjádření pro výpočet nových souřadnic bodů je velmi podobné afinním transformacím, avšak využívá i třetí řádek transformační matice. Projektivní transformace zachovává přímky, ale nezachovává jejich rovnoběžnost. Transformační matice vypadá takto:

$$K * \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{15}$$

Matice (15) obsahuje devět parametrů s tím, že parametr *i* je konstantou o hodnotě 1. Matice má proto 8 stupňů volnosti. Principem je znát alespoň čtyři dvojice bodů, které si budou navzájem odpovídat ve vstupním/výstupním obrázku. V našem případě se jedná o nalezené souřadnice rohů detekovaného objektu, s tím, že si určíme, kde se budou tyto body nacházet ve výstupním obrázku. Polohu rohů ve výstupním obrázku zvolíme tak, že roh nejblíže k počátku souřadnicového systému bude mít souřadnice [0;0] a ostatní zvolíme podle toho, jaký rozměr, resp. tvar výstupního obrázku požadujeme. Obecná forma projektivního mapování bodů vypadá následovně:

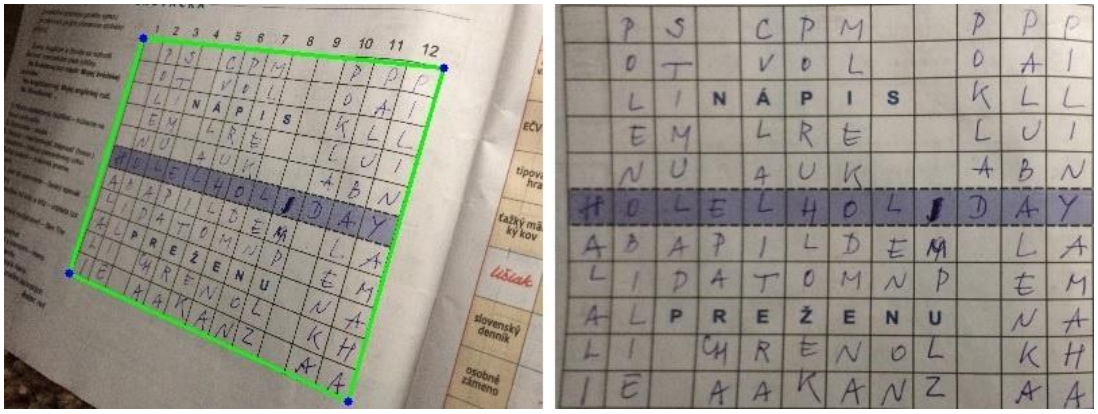
$$x_k = \frac{au_k + bv_k + c}{gu_k + hv_k + 1} \quad \Rightarrow \quad u_k a + v_k b + c - u_k x_k g - v_k x_k h = h_k \tag{16}$$

$$y_k = \frac{du_k + ev_k + f}{gu_k + hv_k + 1} \quad \Rightarrow \quad u_k d + v_k e + f - u_k y_k g - v_k y_k h = y_k \tag{17}$$

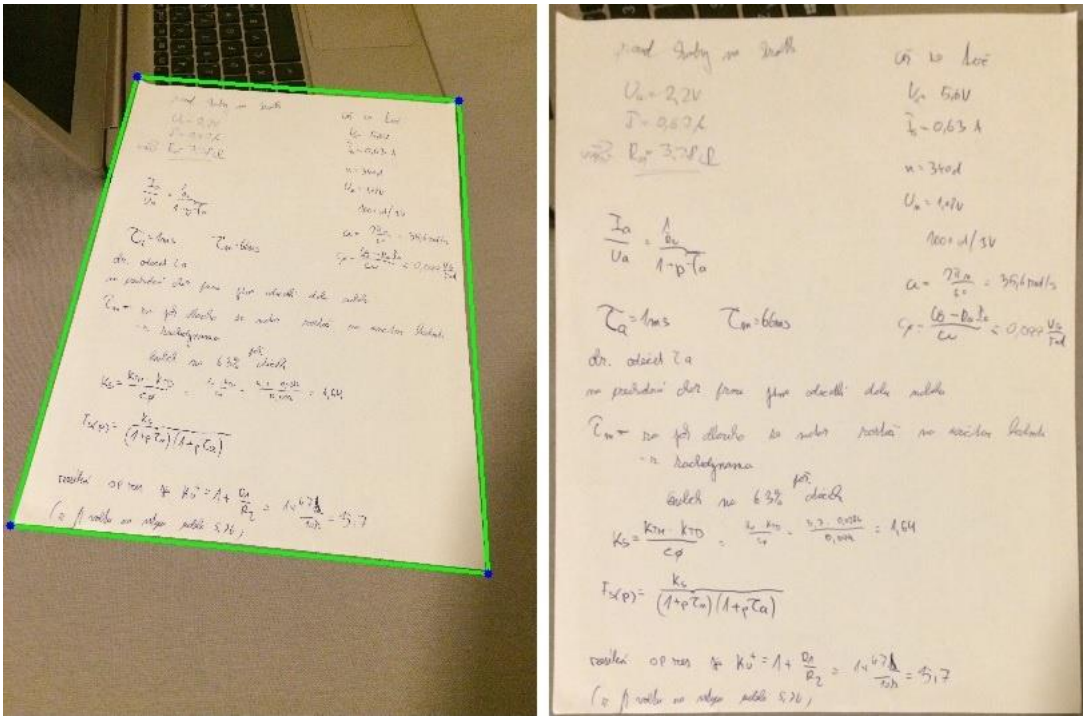
Kde hodnoty u_k a v_k jsou hodnoty souřadnic rohů detekovaného objektu a hodnoty x_k a u_k jsou hodnoty souřadnic, na které se mají jednotlivé rohy namapovat. Tyto rovnice se dají dále zapsat ve formě systému o velikosti 8x8 (18). Vyřešením tohoto systému dostaneme hodnoty osmi neznámých, které značí parametry transformační matice.

$$\begin{bmatrix} u_0 & v_0 & 1 & 0 & 0 & 0 & -u_0x_0 & -v_0x_0 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -v_1x_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2x_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -v_3x_3 \\ 0 & 0 & 0 & u_0 & v_0 & 1 & -u_0y_0 & -v_0y_0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1y_1 & -v_1y_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2y_2 & -v_2y_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3y_3 & -v_3y_3 \end{bmatrix} * \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{18}$$

Pro výpočet těchto parametrů si vypomůžeme funkcí *getPerspectiveTransform*, do které vložíme, jak vstupní, tak výstupní souřadnice rohů. Pro účely tohoto programu zvolíme rohy výstupu tak, aby výsledná velikost obrázku byla shodná s velikosti původního vstupního snímku a abychom zachovali daný poměr. Následně použijeme funkci *warpPerspective*, která provede operaci geometrické transformace se vstupním obrázkem a transformační matici, kterou jsme vypočetli v předešlém kroku. Výsledkem je obrázek objektu, který už neobsahuje přebytečné pozadí a pohled je kolmý na rovinu, na které se nachází.



Obrázek 3-20: Využití projektivní transformace pro opravu perspektivy detekovaného objektu z kategorie sudoku_crosswords



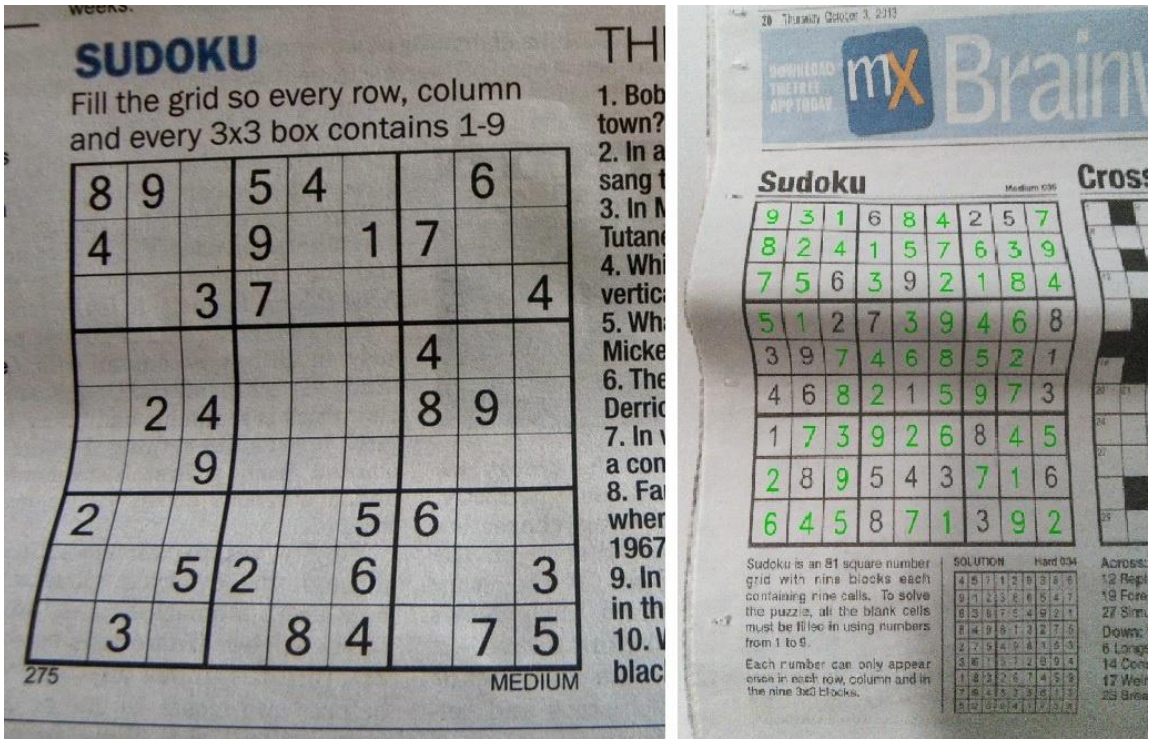
Obrázek 3-21: Využití projektivní transformace pro opravu perspektivy detekovaného objektu z kategorie paper_documents

U obrázků z kategorie paper_documents může být ještě vhodné po opravě perspektivy ořezat okraje tak, abychom se zbavili případných zbytků pozadí a zároveň neodstranili důležité informace na papírech.

3.8 Oprava mřížky u objektů typu sudoku

U objektů typu sudoku z kategorie sudoku_crosswords se dále nabízí oprava, resp. vyrovnání jejich mřížek. Mřížka jednotlivých hlavolamů sudoku, vytištěných na papíře, totiž častokrát podléhá deformacím. Vzniknuté deformace lze přisoudit nerovnostem roviny papíru. Jedná se především o zvlnění, kde papír dostatečně nepřiléhá na podložku, na které leží.

Tyto deformace mají za následek ohýbání čar, které tvoří danou mřížku. Příklad obrázků sudoku se zdeformovanou mřížkou můžeme vidět na obrázku 3-22.



Obrázek 3-22: Snímky typu sudoku se zdeformovanou mřížkou

Pro vyřešení tohoto problému zvolíme podobné řešení, jako při prvotní detekci objektu a následné opravy perspektivy. Obrázek si nejprve předpřipravíme opravou perspektivy za pomoci čtyř rohových bodů. Souřadnice bodů si ale posuneme o určitou velikost ve směru nejbližšího rohu obrazu. Tímto krokem zaručíme, že pohled na papír bude přibližně kolmý na rovinu, na které leží. Zároveň zamezíme nechtěnému odseknutí části mřížky, v případě, že jsou okrajové čáry mřížky zvlněné směrem ke krajům obrazu.

Abychom dokázali narovnat mřížku sudoku, je následně potřebné úspěšně detekovat všech 81 dílčích čtverečků, které jako celek tvoří samotný hlavolam sudoku.

Rohy všech čtverečků totiž tvoří body – průsečíky mřížky. Myšlenka opravy mřížky spočívá v opravě perspektivy dílčích čtverečků a jejich následného poskládání do výstupního obrazu. Pro zajištění co největší šance detekce je výhodné dále odstranit přebytečné stíny nacházející se v obrazu. O této problematice bude detailně pojednávat kapitola 3.9.

Najít kontury všech čtverečků ze vstupního obrazu jako celku, je ale poměrně náročné. Lepším řešením je hledat kontury v malých částech obrazu. Pro tento případ využijeme zjištěné souřadnice detekovaného objektu. Pro zjištění přibližné délky strany jednoho čtverečku provedeme operaci (*TRx* - *TLx*/9), kde *TRx* značí hodnotu souřadnice pravého horního rohu ve směru x a *TLx* pak hodnotu souřadnice levého horního rohu ve směru x. V případě, že se nejedná o čtverec, ale obdélník, analogicky se vypočítá druhá strana čtyřúhelníku.

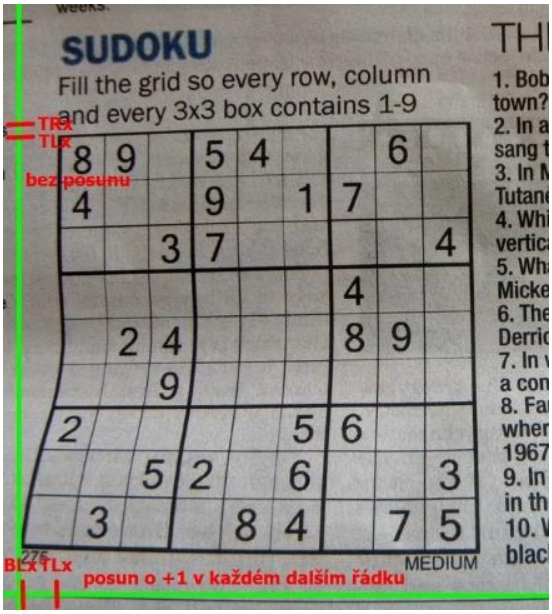
Jako počáteční bod výseku obrazu nám bude sloužit levý horní roh objektu. Rozměr výseku pro umožnění detekce kontur, ale musí být o něco větší, než samotný rozměr čtyřúhelníku. Mřížka sudoku může být navíc zdeformována v různých směrech, proto je vhodné zvolit koeficient zvětšení, o který bude výsledná délka výseku v obou směrech větší, než samotná délka čtyřúhelníku. V programu se osvědčila startovací hodnota, kde koeficient zvětšení představoval třetinu délky čtyřúhelníku v příslušném směru. Tato velikost výseku a následný posun v obrázku vždy o stejnou hodnotu délky výseku funguje pro většinu čtyřúhelníků v detekovaném objektu. Z důvodu deformací, se ale může stát, že daný rozměr výseku nemusí obsáhnout celou plochu hledaného čtyřúhelníku. Pro tento případ je vhodné detekci provádět za použití více hodnot zvětšení. Na obrázku 3-23 vidíme, že po aplikování základního koeficientu zvětšení, v některých případech neobsáhneme ve výseku obrazu celý čtyřúhelník. Postupným zvětšováním rozměrů výseku se ale nakonec dostaneme k celému obrysu hledané oblasti.

Obrázek 3-23: Výseky v obrazu pro detekci čtyřúhelníku při použití různých koeficientů zvětšení

Některé obrázky sudoku jsou navíc vyfotografované tak, že čáry tvořící mřížku nejsou rovnoběžné s okrajem obrázku, jak ve směru x pro horizontální čáry, tak ve směru y pro vertikální čáry. To má za následek to, že se postupným posouváním výseku dostaneme mimo právě analyzovaný řádek, resp. sloupec v mřížce. Abychom předešli tomuto nežádoucímu stavu, je vhodné do parametrů pro výpočet rozměru výseku zařadit koeficienty posunutí ve směru osy x a y. Hodnota koeficientu pro posunutí ve směru x se

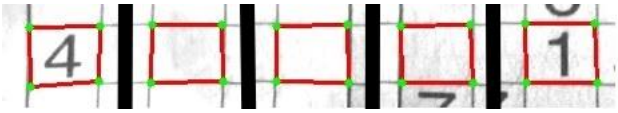
vypočte, jako rozdíl y-ové souřadnice levého horního rohu a y-ové souřadnice pravého horního rohu. Podobně u koeficientu pro směr y využijeme x-ovou souřadnici levého horního rohu a odečteme od ni x-ovou souřadnici levého dolního rohu. Oba výsledky pak ještě podělíme číslem 9 – jeden řádek a sloupec obsahuje 9 dílčích čtyřúhelníků. Protože u diskrétních obrazů pracujeme s přirozenými čísly, hodnoty koeficientů zaokrouhlíme.

Je potřeba přiznat, že toto řešení nemusí vždy fungovat spolehlivě. Čáry mřížky totiž mohou být zvlněné značně nepravidelně. Existují případy, kde ke zvlnění dojde ve středu čáry, s tím, že oba konce čáry leží na stejné přímce. V takové situaci navržený algoritmus přiřadí koeficientu posunu nulovou hodnotu. Zůstává tedy spoléhat na postupné zvětšování výseku obrazu, jak bylo popsáno výše. Zvětšování výseku ovšem nelze provádět donekonečna. Při velmi velkém rozměru výseku, dostaneme obraz, ve kterém se nachází více čtyřúhelníků z mřížky sudoku a tím pádem algoritmus nemusí detekovat ten správný.



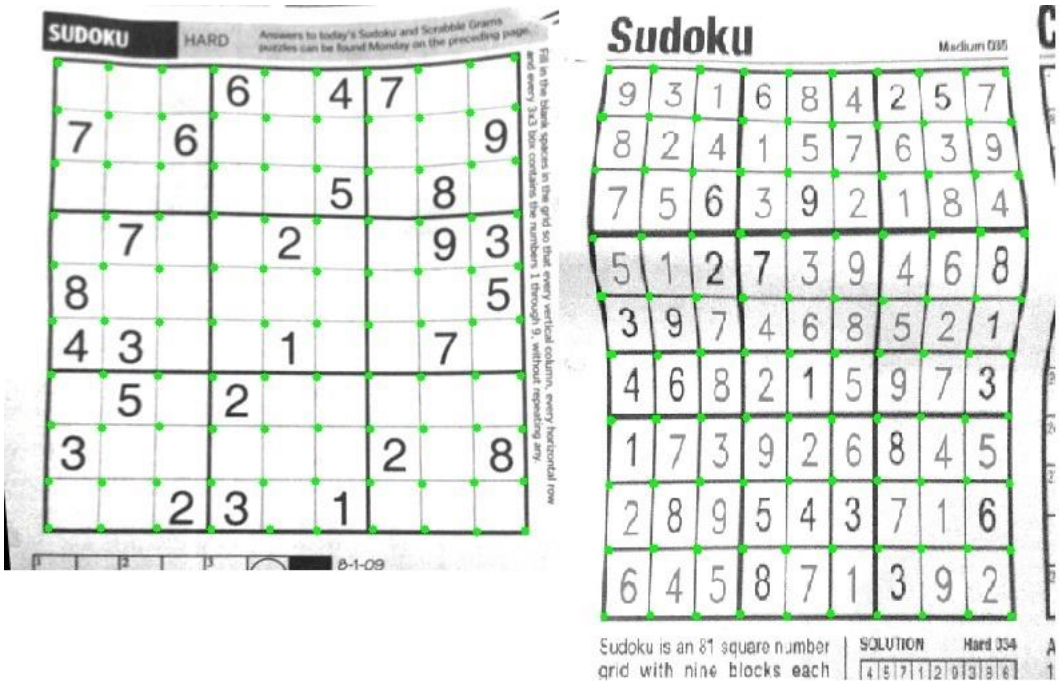
Obrázek 3-24: Znázornění výpočtu posunu v obou směrech

Samotná detekce se vykoná nejprve použitím adaptivního prahování pro obdržení obrazu hran (popis v kapitole 3.5.3) a následným hledáním kontur, které jsme detailně rozebírali v kapitole 3.6.1. Je důležité, abychom u obrazu hran dokázali vždy najít spojité kontury hledaných čtyřúhelníků. To docílíme otestováním více kombinací parametrů pro adaptivní prahování. Pokud se nám i přesto nepovede detekovat hledanou oblast, pomůžeme si použitím morfologických operací, konkrétně operací morfologického uzavření a následné dilatace. Tyto operace jsme podrobně popisovali v kapitole 3.4.



Obrázek 3-25: Detekce čtyřúhelníků ve výsecích obrazu v různých místech analyzované mřížky

Za detekovaný čtyřúhelník považujeme oblast, jejíž konturu lze aproximovat na útvar se čtyřmi vrcholy. Zároveň se musí hodnota obsahu analyzované oblasti nacházet v rozmezí od 60 do 140% předpokládaného obsahu jednoho čtverečku – čtyřúhelníku. Tento referenční obsah vypočítáme vynásobením délek čtyřúhelníku v obou směrech, které jsme už předem zjistili ze souřadnic rohů. Po úspěšné detekci nakonec převedeme nalezené souřadnice bodu mřížky z výseku tak, aby odpovídaly skutečné poloze bodu ve vstupním obrazu. Příklady úspěšné detekce všech bodů mřížky vidíme na obrázku 3-26.

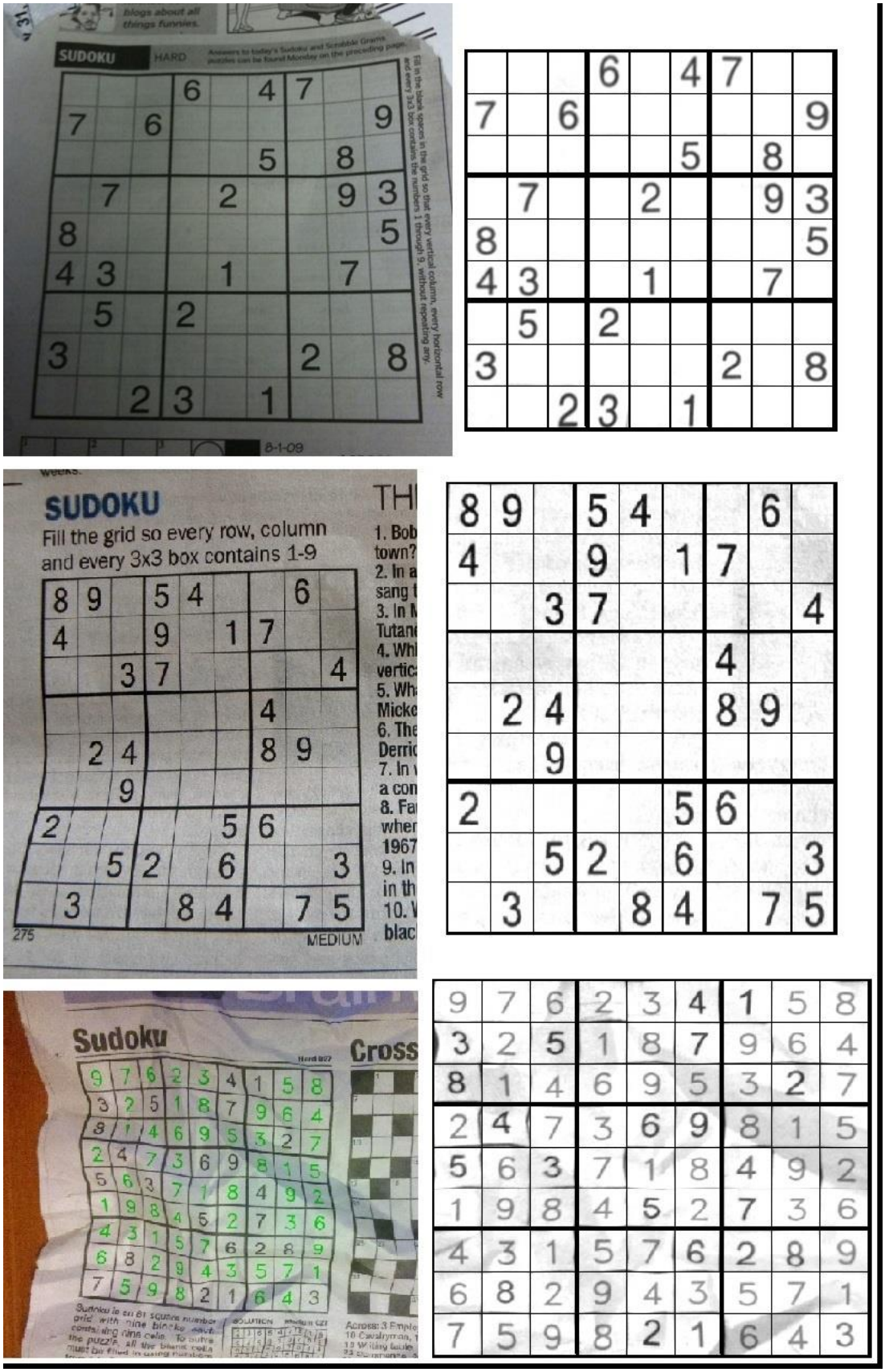


Obrázek 3-26: Příklady detekce jednotlivých bodů mřížky

U některých obrázků se kvůli zvlněním a nedostatečné velikosti výseku stává, že algoritmus detekce nenajde hledanou oblast. Tento problém je možné řešit využitím souřadnic ze sousedního bodu v mřížce. Od souřadnice sousedního bodu napravo ve směru x odpočítáme přibližnou hodnotu délky čtyřúhelníku v tom stejném směru. Souřadnice ve směru y zůstane nezměněna. Pokud se jedná o body, které se v řádcích nachází nejvíc vpravo, využijeme stejným způsobem sousední body vlevo. Jediným rozdílem je, že k souřadnici ve směru x přičteme hodnotu délky čtyřúhelníku.

Toto aproximační řešení se dá považovat jenom za kompromisní a dobře funguje jenom při malých ohybech jednotlivých čar mřížky. V případech, kdy se nepovede najít více sousedících bodů za sebou, už nelze považovat detekci bodů za úspěšnou a uživatel se musí spokojit se základní opravou perspektivy pomocí čtyř rohových bodů mřížky sudoku.

Poté, co obdržíme souřadnice všech průsečíků mřížky, provedeme opravu perspektivy jednotlivých čtyřúhelníků tak, jak je vysvětleno v kapitole 3.7. Protože detekce čtyřúhelníků probíhá tak, že se hledají kontury prostoru ohraničeného čárami mřížky, je vhodné tyto souřadnice posunout o pár pixelů. Tím docílíme polohy bodů přímo v prostoru, kde se střetávají jednotlivé čáry. Nakonec upravené čtyřúhelníky poskládáme do výsledného obrázku. Pro „zkrášlení“ výsledku překreslíme čáry, které ohraničují všech 9 větších čtverců sudoku pomocí knihovní funkce *line*. Výsledkem jsou, jak znázorňuje obrázek 3-27, obrázky sudoku s vyrovnanou mřížkou bez nepotřebného pozadí.

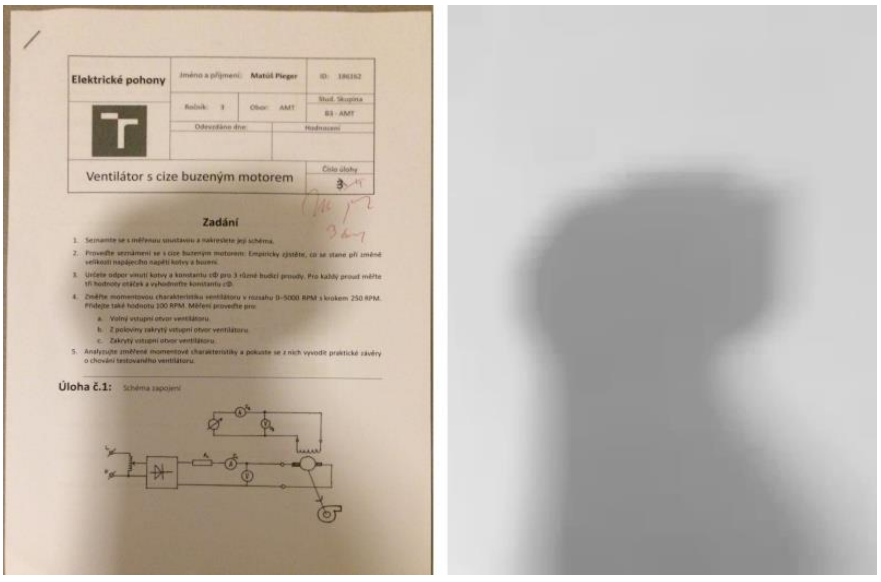


Obrázek 3-27: Obrázky typu sudoku s opravenou mřížkou

3.9 Odstranění stínů

Po úspěšné opravě perspektivy a dalších geometrických zkreslení přejdeme k operaci odstranění přebytečných stínů. Stíny na detekovaném objektu vznikají z různých příčin. Mezi nejčastější typy stínů, patří stín vytvořený samotným fotoaparátem v závislosti na směru, ze kterého světlo přichází. To se týká i jiných překážek v okolí, které mohou bránit světlu v průchodu prostorem. Stíny způsobuje taky nerovnoměrné osvětlení, nebo nedostatek osvětlení v části obrazu. Podrobně se této problematice věnuje kapitola 2.1.1.

Pro odstranění stínů potřebujeme najít způsob, kterým nechtěné stíny izolujeme od zbytku obrazu. K tomu využijeme morfologické operace, podrobně popsané v kapitole 3.4. Nejlepší volbou je výběr morfologického uzavření. To nám vyhladí text a zároveň ponechá analyzované stíny. Důležitým aspektem je ale správný výběr a velikost strukturního elementu. V programu se osvědčilo používání strukturního elementu typu čtverec, který tvoří matici o hodnotách 1. Pokud použijeme element o příliš malém rozměru, větší objekty, jako je třeba logo VUT na obrázku 3.28, zůstanou zachované. To se děje z důvodu používání dilatace na šedotónovém obrázku. Ta nejprve sečte strukturní element se zpracovávanou části obrazu a následně hledá maximum výsledné matice. Při příliš malém rozměru proto operace morfologického uzavření nemá šanci vyhladit větší objekty tak, aby splynuly s papírem. V programu se využívá rozměr strukturního elementu o velikosti 101x101. Takový rozměr vyhladí i větší objekty v obrazu a současně neporuší hledané stíny. Nevýhodou je výpočetní náročnost.



Obrázek 3-28: Segmentace nechtěného stínu ve zpracovávaném snímku

Vyhlazení obsahu papíru je důležité, pro následující krok při odstraňování stínů. V tomto kroku využijeme aritmetické operace, které je možné s diskrétními obrazy,

jakožto maticemi o nějakém rozměru, provádět. Konkrétně využijeme operaci dělení zpracovávaného obrazu s obrazem, který obsahuje oddělený stín. Každý bod v obrazu se tedy vydělí s příslušným bodem v obrazu stínu. To znamená, že hodnoty všech bodů znázorňující stín zůstaly totožné ve vysegmentovaném obrazu. Při dělení se výsledné hodnoty budou pohybovat v intervalu od 0 do 1. V místech, kde se nachází stín, budou výsledné hodnoty rovné 1. Naopak, jelikož jsme v předešlém kroku vyhladili text a celkově obsah papíru, dostaneme po vydělení příslušných bodů hodnoty o málo vetší než hodnota 0. Pokud bychom použili v předešlém kroku příliš malý strukturní element, větší objekty by se promítly i do segmentovaného obrazu a po dělení bychom v těchto místech dostali hodnoty blízké 1. Tyto objekty by se tím pádem nepromítly do výstupního obrazu. To se děje, protože při reprezentaci jasových úrovní, jsou hodnoty tmavších jasových úrovní menší, než hodnoty světlejších jasových úrovní. Maximum našeho intervalu bude proto značit bílou barvu a minimum černou barvu. Tyto hodnoty musíme pak dále upravit tak, abychom dokázali zobrazit obrázek v standardní osmi-bitové hloubce. Z toho důvodu je potřebné znát pojem histogram, o kterém pojednává následující podkapitola.

3.9.1 Histogram a jeho normalizace

Histogram je grafické znázornění četnosti jasových úrovní v obrazu na hodnotách těchto úrovní. Funkční hodnoty jednotlivých úrovní vypočítáme jako:

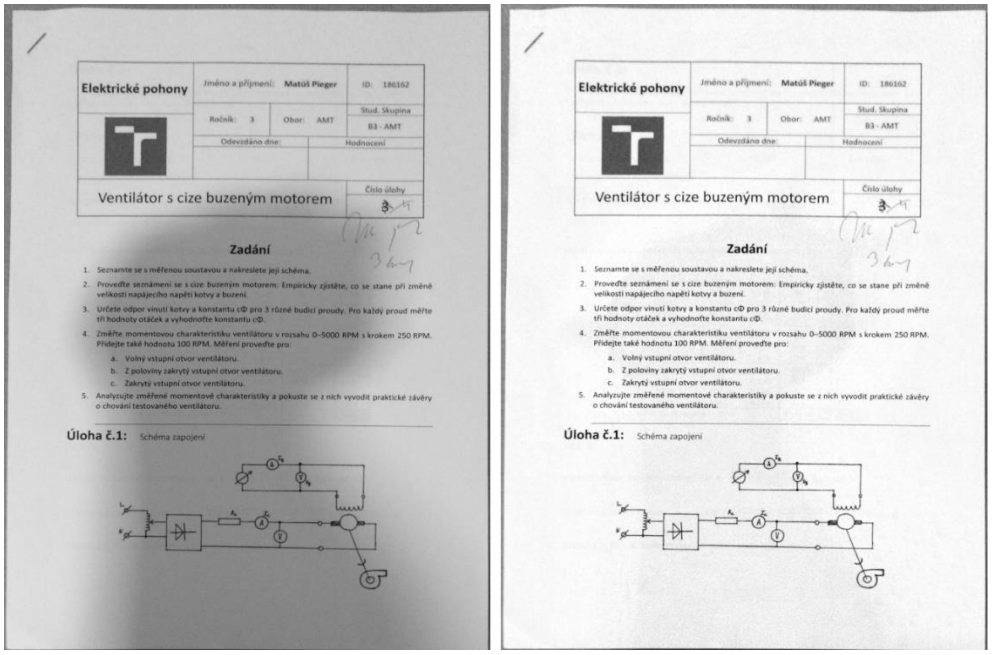
$$h[i] = \frac{počet\ bodů\ s\ určitou\ jasovou\ úrovní}{celkový\ počet\ bodů} \tag{18}$$

V osmi-bitové hloubce využíváme 256 jasových úrovní obrazu. Naše výstupní hodnoty po dělení dvou diskrétních obrazů jsou ale v intervalu od 0 do 1 a je nutné je přepočítat tak, aby každá z hodnot ležela v intervalu od 0 do 255. Tento krok se nazývá normalizace histogramu (19). Cílem normalizace je dosáhnout využití celého rozsahu jasových úrovní dané bitové hloubky. Výpočet nových hodnot provedeme pomocí vzorce:

$$y = f(x) = 255 * \frac{x - x_{min}}{x_{max} - x_{min}} \tag{19}$$

V našem případě bude hodnota x_{max} rovná 1 a x_{min} rovná 0. Proměnná x pak značí aktuální hodnotu jasové úrovně, kterou normalizujeme. Pro normalizaci využijeme v programu funkci *normalize*. Výsledkem je obraz zbavený přebytečných stínů.

Takový obraz ale pořád může obsahovat drobné stíny, které způsobují, že se nám plocha papíru nebude jevit úplně čistě bílá. Tento nedostatek vyřešíme prahováním na základě jasových hodnot.



Obrázek 3-29: Odstranění nejvýraznějších stínů použitím morfologických operací a normalizace histogramu

3.9.2 Prahování na základě jasových hodnot

Prahování je funkce, která upravuje hodnoty jasových úrovní bodů ve vstupním obrazu podle daného předpisu. Prahování se častokrát používá pro binarizaci obrazu, kde všechny hodnoty nad prahem se nastaví na hodnotu maxima dané bitové hloubky a všechny hodnoty pod prahem na minimum.

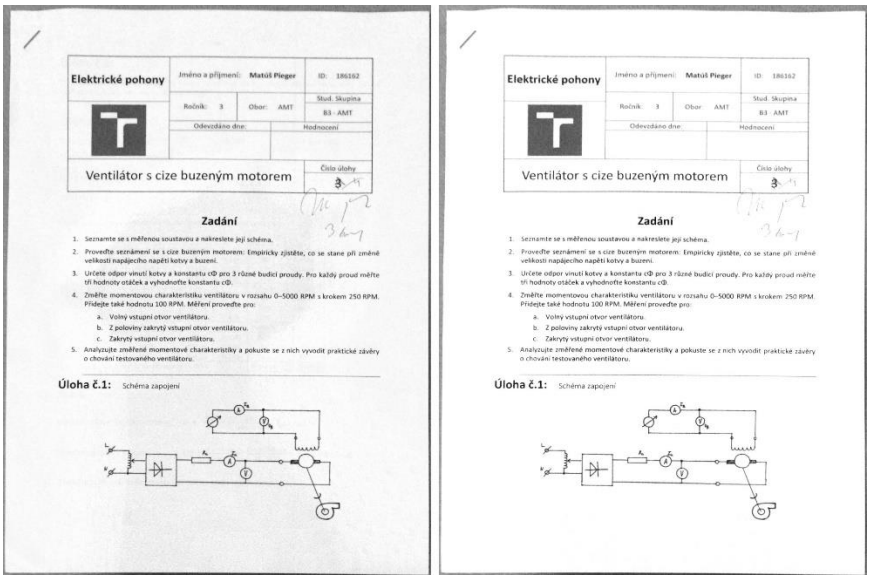
Našim cílem je potlačit zbylé drobné stíny tak, aby všechny plochy v papíru, které neobsahují text, resp. nejsou popsané, byly vyobrazené bílou barvou (hodnota jasové úrovně 255). Hodnota prahu, která splňuje tyto požadavky, se pohybuje v rozmezí hodnot 200 až 255. V programu využijeme funkci *threshold*, která nabízí více druhů prahování. Pro náš případ je nejlepší použít typ *thresh_trunc*. Tento typ prahování potlačuje všechny jasové hodnoty bodů nad práh tak, že je změní na hodnotu prahu. Hodnoty pod prahem ponechá nezměněné. Prahovací funkce bude vypadat následovně:

$$výstup(x,y) = \begin{cases} práh & vstup(x,y) > práh \\ vstup(x,y) & vstup(x,y) \leq práh \end{cases} \quad (20)$$

To znamená, že všechny hodnoty jasových úrovní bodů, které mají hodnotu větší než práh, se v upraveném obrázku změní na hodnotu prahu. Ostatní body s jasovou hodnotou, která je menší než práh, tedy popsané místa v papíru, se nezmění.

Při testování programu se nejvíc osvědčila hodnota prahu rovná 230. Po aplikací prahování, je ale navíc nutné znovu znormalizovat histogram tak, abychom využívali celý rozsah jasových úrovní. Těmito kroky nakonec docílíme, že všechny místa obsahující

drobné stíny, jsou nově vyobrazené bílou barvou, tedy přirozenou barvou bílého papíru. Výhodou prahování je další zlepšení čitelnosti obrazu a šetření inkoustu při případném tisku výsledného snímku. Porovnání obrázků před a po prahování vidíme na obrázku 3.30 a 3.31.



Obrázek 3-30: Odstranění zbylých drobných stínů pomocí prahování jasových úrovní

Obrázek 3-31: Odstranění stínů u obrázku z kategorie sudoku_crosswords

3.10 Redukce bitové hloubky obrazu

Posledním krokem v úpravě vstupních snímků je vytvoření algoritmu pro redukci počtu jasových úrovní v šedotónových obrázcích, dle volby uživatele. Tato operace má význam v případě šetření inkoustu při tisku výsledného snímku, zároveň ale zhoršuje jeho kvalitu. Bude tedy na uživateli, jaký kompromis zvolí.

Redukce počtu jasových úrovní souvisí s pojmem bitová hloubka obrazu. Jak už bylo naznačeno v předešlých kapitolách, každý obraz je vytvořen z bodů o určitých hodnotách jasových úrovní. Bitová hloubka vyjadřuje, kolik různých jasových úrovní je možné v obrazu použít. Z toho vyplývá, že čím větší bitovou hloubku obraz obsahuje, tím lepší bude vizuální dojem z vyobrazení barev.

Obrázky ze vstupních datasetů používaných v této práci pracují s osmi-bitovou hloubkou obrazu. To znamená, že počet jasových úrovní, které využívají je rovný 2^8, resp. 255 úrovním. Používáním osmi-bitové hloubky dostáváme obrazy v dobré kvalitě, na druhou stranu je takový počet oproti menším bitovým hloubkám náročnější na paměť. V našem případě, kdy zpracováváme papírové dokumenty, může být osmi-bitová hloubka zbytečná. Tyto dokumenty obsahují obvykle tmavý text nebo tabulku na světlém, resp. bílém pozadí. Proto může mnohokrát pro dostatečnou kvalitu stačit použití menší bitové hloubky nebo dokonce jenom dvou barev (černá a bílá). Abychom byli schopní snížit počet jasových úrovní v obrazu, využijeme jasové transformace a tzv. Look-up tabulku.

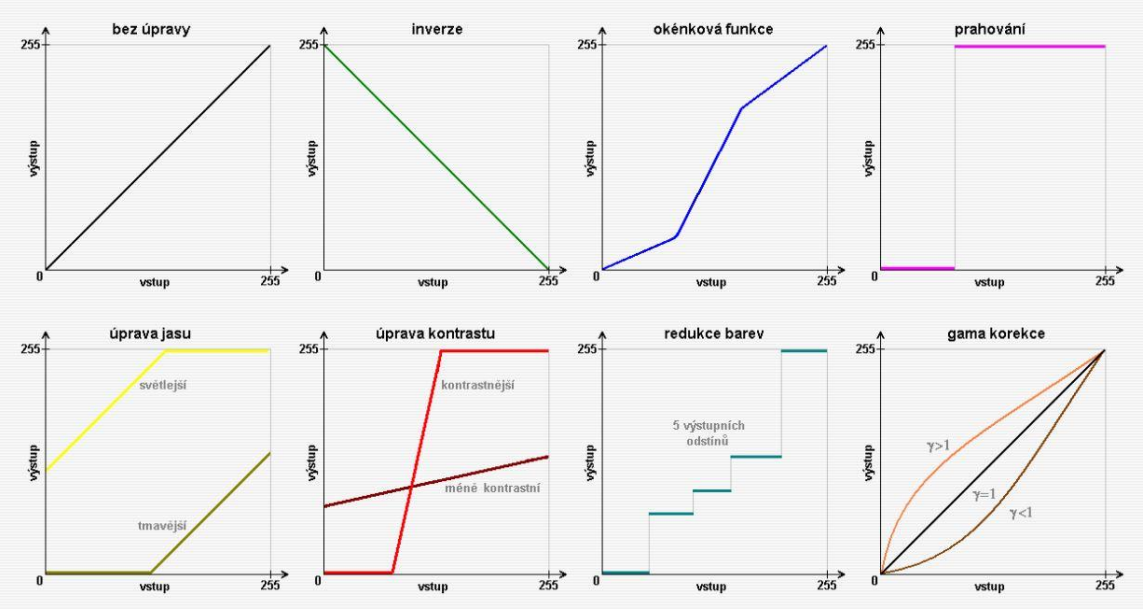
3.10.1 Redukce barev pomoci Look-up tabulky

Pro pochopení pojmu Look-up tabulka (LUT), je potřebné ozřejmit si smysl jasových transformací. Jasové transformace slouží pro změnu hodnot obrazové funkce vstupního obrazu podle daného pravidla T (21).

$$obraz\ I, MxN \Rightarrow [\ \ T\ \ ] \Rightarrow obraz\ J, MxN \tag{21}$$

Je možné je rozdělit do tři úrovní: globální, lokální a bodové. Mezi jasové transformace, které jsme již v práci využívali, patří potlačení šumu a hledání hran. Jedná se o lokální jasové transformace a nová hodnota každého pixelu se počítá z hodnot jeho okolí. Globální transformace, jak název napovídá, počítají novou hodnotu pixelu z celého obrazu. Do této kategorie patří např. Fourierova transformace obrazu, v práci se ale nepoužívá. Posledním typem jasových transformací jsou bodové jasové transformace, mezi které patří kromě jiných i normalizace histogramu. V tomto případě se nová hodnota pixelu počítá pouze z příslušného pixelu ve vstupním obrazu.

Tento typ transformací se používá nejčastěji, a jako pravidlo T se využívají převodní funkce. V diskrétním pojetí obrazu se jedná o vyhledávací, resp. LUT tabulku. Tuto tabulku si lze představit jako datovou strukturu, kde pro každou možnou jasovou úroveň je určená hodnota, na kterou se bod s danou jasovou úrovní transformuje. Pro osmi-bitové obrázky bude vektor vstupu představovat hodnoty od 0 do 255 s krokem 1. Výstupem bude vektor o stejné velikosti, hodnoty tohoto vektoru budou záviset na typu převodní funkce. Obrázek 3-32 znázorňuje některé typy převodních funkcí.



Obrázek 3-32: Vybrané typy převodních tabulek

Vidíme, že vhodnou volbou převodní charakteristiky lze v obrazu provést různé typy úprav, jako je např. změna jasu a kontrastu obrazu nebo prahování obrazu, které jsme využívali při odstraňování přebytečných stínů. V tomto případě má pro nás význam použít převodní funkci typu redukce barev.

Tato funkce nejprve rozdělí vstupní vektor všech možných jasových hodnot na stejně dlouhé úseky v závislosti na tom, kolik výstupních jasových úrovní požadujeme. Pokud tedy chceme např. dvou-bitovou hloubku, počet barev, resp. jasových úrovní ve výstupním obrázku bude rovný 2^2, což jsou čtyři barvy. To znamená, že se vstupní vektor rozdělí na čtyři stejně dlouhé úseky.

Následně je potřeba pro každý úsek určit právě jednu hodnotu jasové úrovně, která se přiřadí všem jasovým úrovním v daném úseku. Tyto hodnoty se získají z mezí jednotlivých úseků. Protože je počet mezí vždy lichý a počet jasových úrovní pro všechny bitové hloubky sudý, rozdělí se vstupní vektor na dvě poloviny. Pro úseky v intervalu <0;126> se využijí spodní meze jednotlivých úseků. Pro úseky v intervalu <127;255> pak horní meze. Tím se docílí rovnoměrného rozložení jasových úrovní ve výstupním obrazu o počtu, který určuje zvolená bitová hloubka. Jednotlivé hodnoty do LUT tabulky pro první polovinu intervalu vypočítáme následovně:

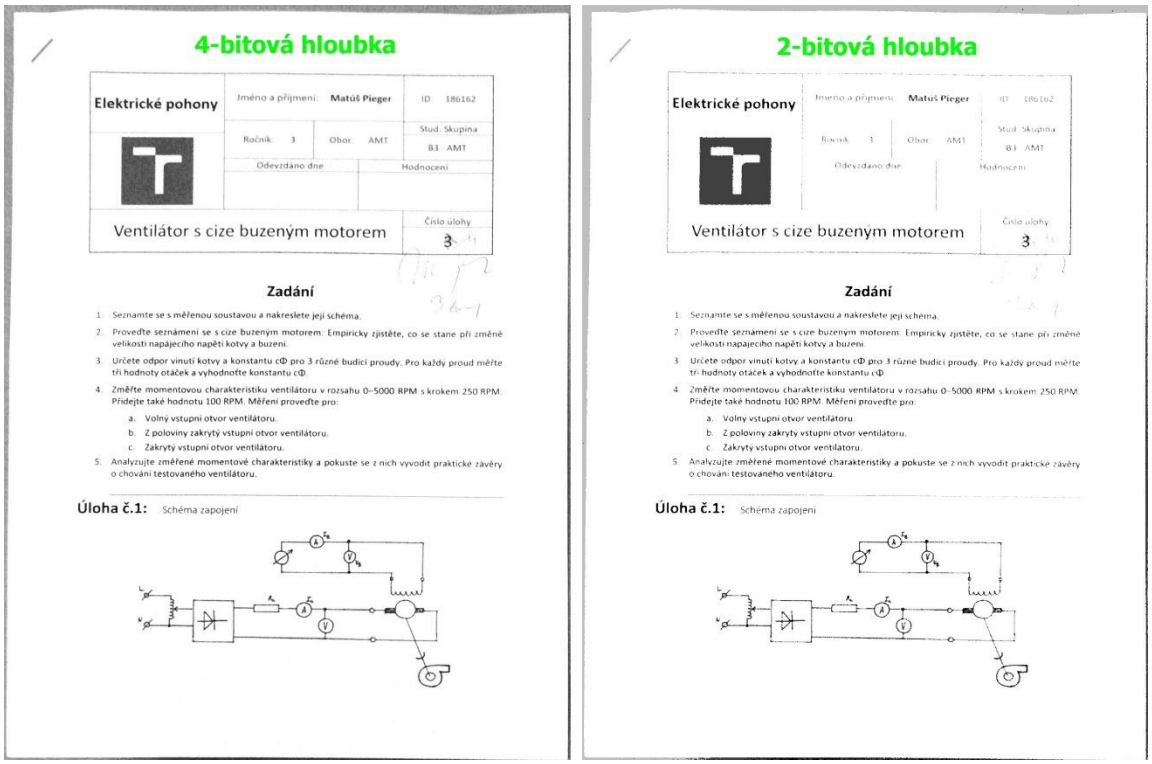
$$LUT[i] = \frac{256}{2^{bitová\_hloubka}} * int\left(\frac{256}{2^{bitová\_hloubka}} * i\right) \quad (22)$$

Kde proměnná *i* značí hodnotu jasové úrovně a klíčové slovo *int* znamená, že podíl v závorce budeme po zaokrouhlení v programu ukládat jako hodnotu přirozeného čísla. Pro druhou polovinu intervalu použijeme vzorec:

$$LUT[i] = \frac{256}{2^{bitová\_hloubka}} * \left(1 + int\left(\frac{256}{2^{bitová\_hloubka}} * i\right)\right) - 1 \qquad (23)$$

Nakonec každý bod ve zpracovávaném obrázku transformujeme na příslušnou hodnotu jasové úrovně uloženou v tabulce. Příklad obrázku, který byl upraven pro rozličné bitové hloubky, vidíme na obrázku 3-33. Rozdíl je viditelný, např. na hraně loga VUT.

Tento způsob redukce bitové hloubky, ale není velmi vhodný pro binarizaci obrázku – převedení obrázku na bitovou hloubku rovnou jedné. Pro tento případ použijeme operaci adaptivního prahování obrazu.



Obrázek 3-33: Porovnání stejného obrázku upraveného na 4-bitovou vs. 2-bitovou hloubku

3.10.2 Adaptivní prahování pro binarizaci obrazu

Při redukci bitové hloubky na úroveň 1 výše popsaný algoritmus rozdělí vektor jasových úrovní v jeho středu, což znamená, že všechny body s jasovou úrovni menší, než 128 budou černé a zbytek bude bílý. Ve vstupních snímcích je ale běžné, že z důvodu různých světelných podmínek, jsou některé části textu nebo čar včetně jejich okolí, vyobrazené body s jasovou úrovní větší nebo naopak menší než hodnota 128. Z toho

důvodu můžeme aplikací algoritmu způsobit nečitelnost, resp. úplné zmizení textu a jiných významných oblastí ve snímku.

Řešením je použití operace adaptivního prahování obrazu. Ta funguje na podobném principu, jako běžné prahování, které jsme rozebírali v kapitole 3.8.2. Odlišuje se tím, že nemá určený jenom jeden globální práh pro celý obraz, ale zvolenou velikosti okolí, rozdělí obraz na mnoho menších části. V těch určuje lokální prahy, na základě kterých pak upravuje hodnoty jednotlivých bodů. Tím se docílí správné rozeznání tmavších bodů od světlejších i v oblastech s neobvyklými světelnými podmínkami. Při použití běžného prahování bychom naopak v příliš tmavých oblastech obrazu dostali po binarizaci čistě černé okolí, v příliš světlých oblastech zase čistě bílé okolí i v případě, že by se na daných místech nacházely pro nás důležité informace. Prahovací funkce adaptivního prahování, kde se hodnota prahu bude lišit pro každé zvolené okolí, bude vypadat následovně:

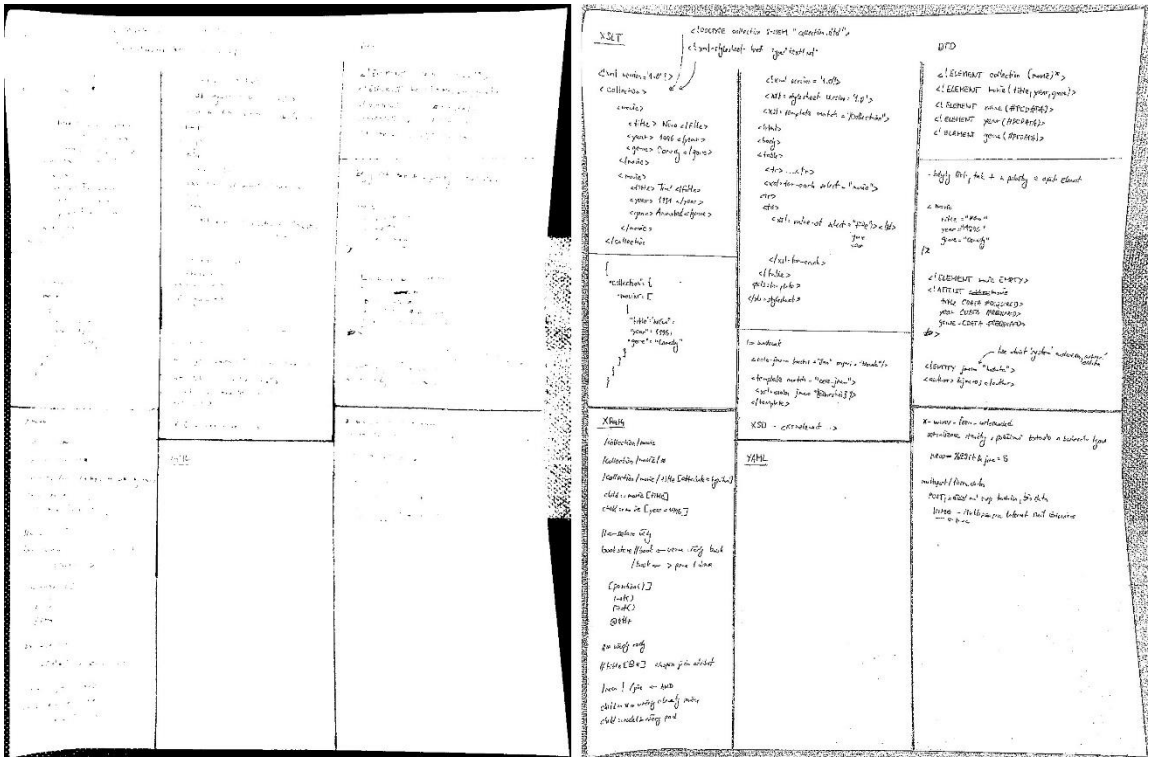
$$výstup(x,y) = \begin{cases} 255 & vstup(x,y) > práh \\ 0 & vstup(x,y) \leq práh \end{cases} \quad (24)$$

V programu využijeme knihovní funkci *adaptiveThreshold*. Pro výpočet lokálních prahů se jako parametr funkce nabízí dvě varianty řešení. První možností je počítat prahy na základě průměru všech hodnot jasových úrovní bodů v analyzovaném okolí. Druhou možnosti je provést vážený součet bodů, kde váhy představuje Gaussova funkce. Po otestování obou variant v programu zjistíme, že o něco lepší výsledky poskytuje řešení součtem a vážením Gaussovou funkcí.

Velikost analyzovaného okolí je lepší volit menší, než větší, v programu použijeme pro kategorii paper_documents velikost okolí 3x3. Pro kategorii sudoku_crosswords je lepší o něco větší okolí, protože i upravované oblasti jsou o něco větší. Použijeme rozměr 13x13. Příliš velké okolí poskytuje horší výsledky především z důvodu, že žádané informace, jako jsou třeba písmena v textu, jsou vyobrazené na relativně malé ploše v obrazu. Pokud proto budeme analyzovat pro výpočet práhu příliš velkou plochu, nemusí se nám povést najít takový práh, který spolehlivě oddělí písmeno od okolí. Na druhou stranu, nevýhodou malého okolí je prahování větších souvislých oblastí, u kterých je vhodné, aby zůstaly tmavé, např. logo VUT z obrázku 3-33. Takové oblasti vždy obsahují menší míru šumu – bodů s vysokou jasovou úrovní. Při analýze malého okolí, proto dochází k určení takových prahů, které obvykle vybělí většinu plochy a ponechají jenom obrysy dané oblasti. Pro takové typy obrázků je vhodnější využít bitovou hloubku větší, než 1, která eliminuje tento nežádoucí jev.

Po aplikaci adaptivního prahování se dostáváme k splnění posledního úkolu této práce, a jak naznačuje i její název, výsledkem je černobílý - binární obrázek zbavený všech nežádoucích vlivů rozebíraných v předešlých kapitolách. Porovnání výsledků po použití

metody, která používá jenom jeden práh a metody adaptivních prahů vidíme na obrázku 3-34.



Obrázek 3-34: Výsledný obrázek po použití jenom jednoho práhu a po použití adaptivního prahování

4 ZHODNOCENÍ VÝSLEDKŮ

V této sekci bude provedena analýza výsledků obou programů.

4.1 Zhodnocení výsledků – Detekce papírových dokumentů

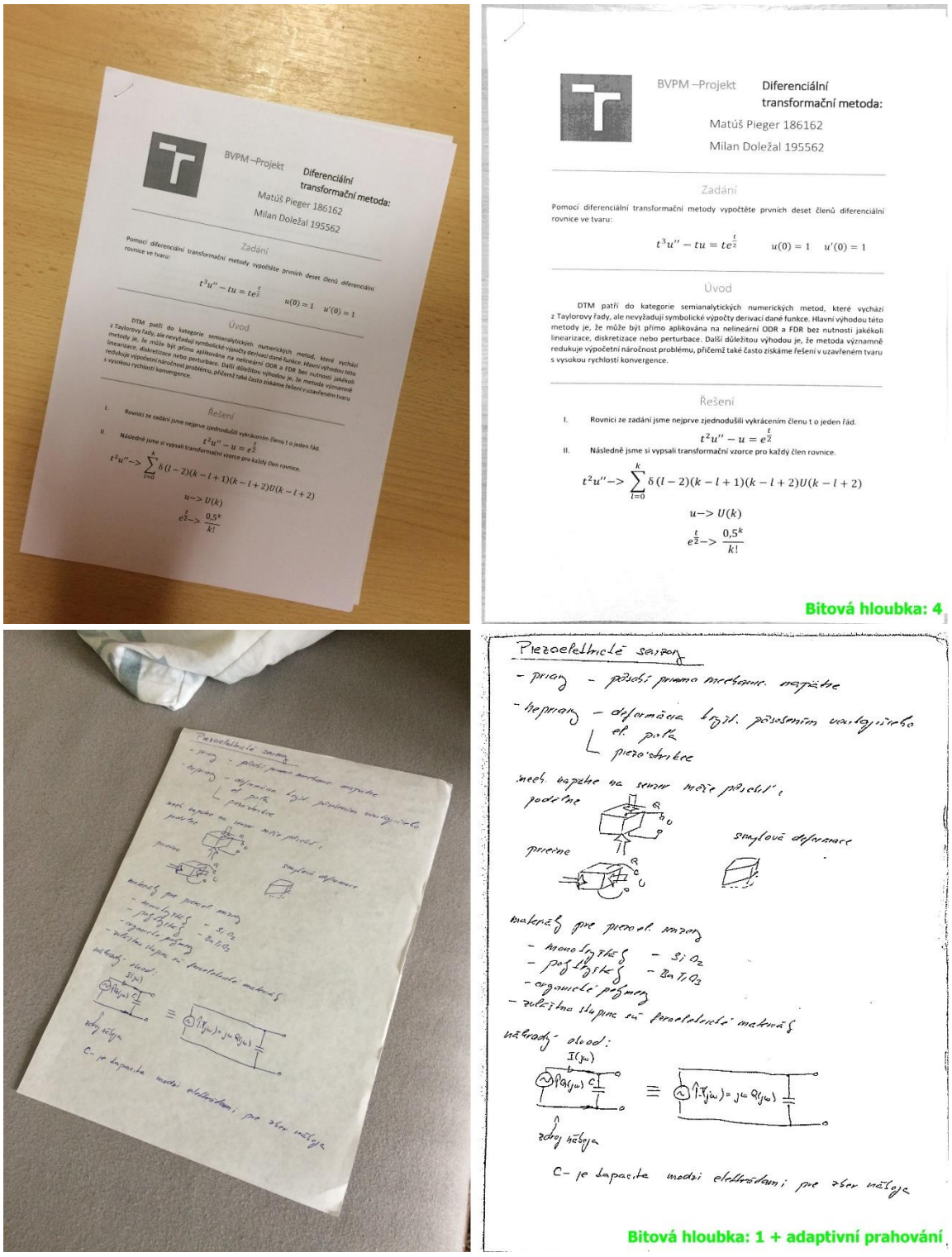
Detekce papírových dokumentů probíhala za pomocí Cannyho detektoru hran. Proto, aby bylo možné zobrazit primárně hrany patřící obrysu hledaného objektu, vypomáhal si program předzpracováním obrazu na základě použití morfologických operací. Tento způsob zaručoval relativní spolehlivost v detekci jednotlivých papírů.

Situace se ovšem měnila v případě, že vstupní snímek obsahoval pozadí, které bylo příliš málo kontrastní vůči samotnému papírovému dokumentu. Jednalo se především o případy příliš světlého, resp. bílého pozadí. To mělo za následek, že nalezené kontury hledaného objektu nebyly spojité a program pak musel spoléhat na nepříliš spolehlivou detekci pomoci přímého hledání rohů. Obecně možno konstatovat, že v takových případech je nejlepší zvolit řešení manuální detekce rohů.

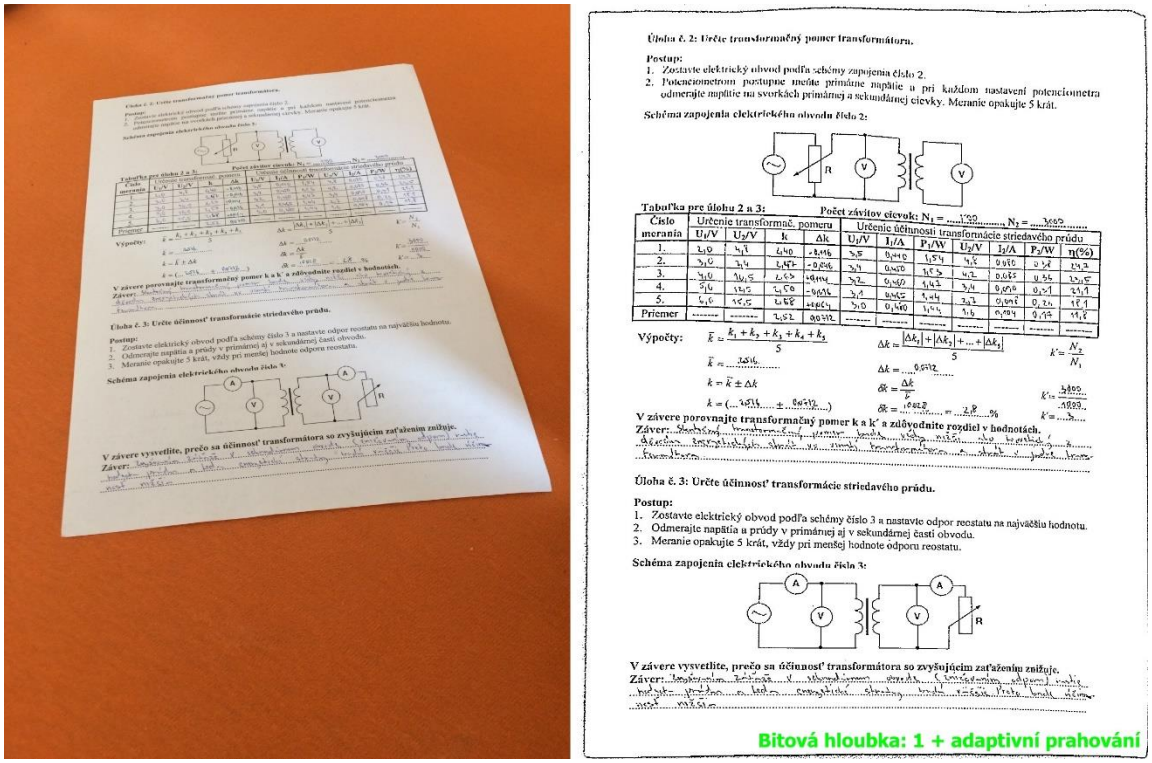
Pro opravu perspektivy program využíval čtyři rohové body nalezené spojité kontury objektu aproximované na čtyřúhelník. Tyto body se následně využily pro výpočet transformační matice a provedení projektivní transformace upravované plochy. Tento algoritmus pracoval bez potíží, jedinou podmínkou bylo, aby papír ve vstupním snímku nebyl vyfotografován s velmi velkou chybou horizontu. Pokud tomu tak bylo, mohlo se stát, že se rohovým bodům nalezené kontury nepřiřadila správná poloha. To znamená, že se například za levý horní roh mohl považovat roh, který byl ve skutečnosti levý dolní atd. Tím pádem se oprava perspektivy neprovedla správně.

S odstraňováním stínů neměl program využitím řešení popsaného v kapitole 3.9. žádné větší potíže. Menší nevýhodou byla větší časová náročnost provedení tohoto algoritmu. Při redukci bitové hloubky se u papírových dokumentů nejvíce osvědčilo používat bitovou hloubku 1, tedy převod na čistě černobílý snímek a zároveň přepnutí na funkci adaptivního prahování. Takové nastavení docílilo nejlepší čitelnost textu. Výhodou bylo i šetření množství barvy při případném tisku výstupních snímků.

V této sekci byla zároveň pro zajímavost vyzkoušena redukce bitové hloubky u několika běžných snímků nezapadajících do kategorie papírových dokumentů. Příklad toho, jak změna bitové hloubky ovlivňuje celkové vyobrazení snímku, vidíme na obrázku 4-3.



Obrázek 4-1: Příklady vstupních a výstupních snímků z kategorie paper_documents



Obrázek 4-2: Příklad vstupního a výstupního snímku z kategorie paper_documents

Obrázek 4-3: Ukázka vlivu změny bitové hloubky na všeobecný obrázek

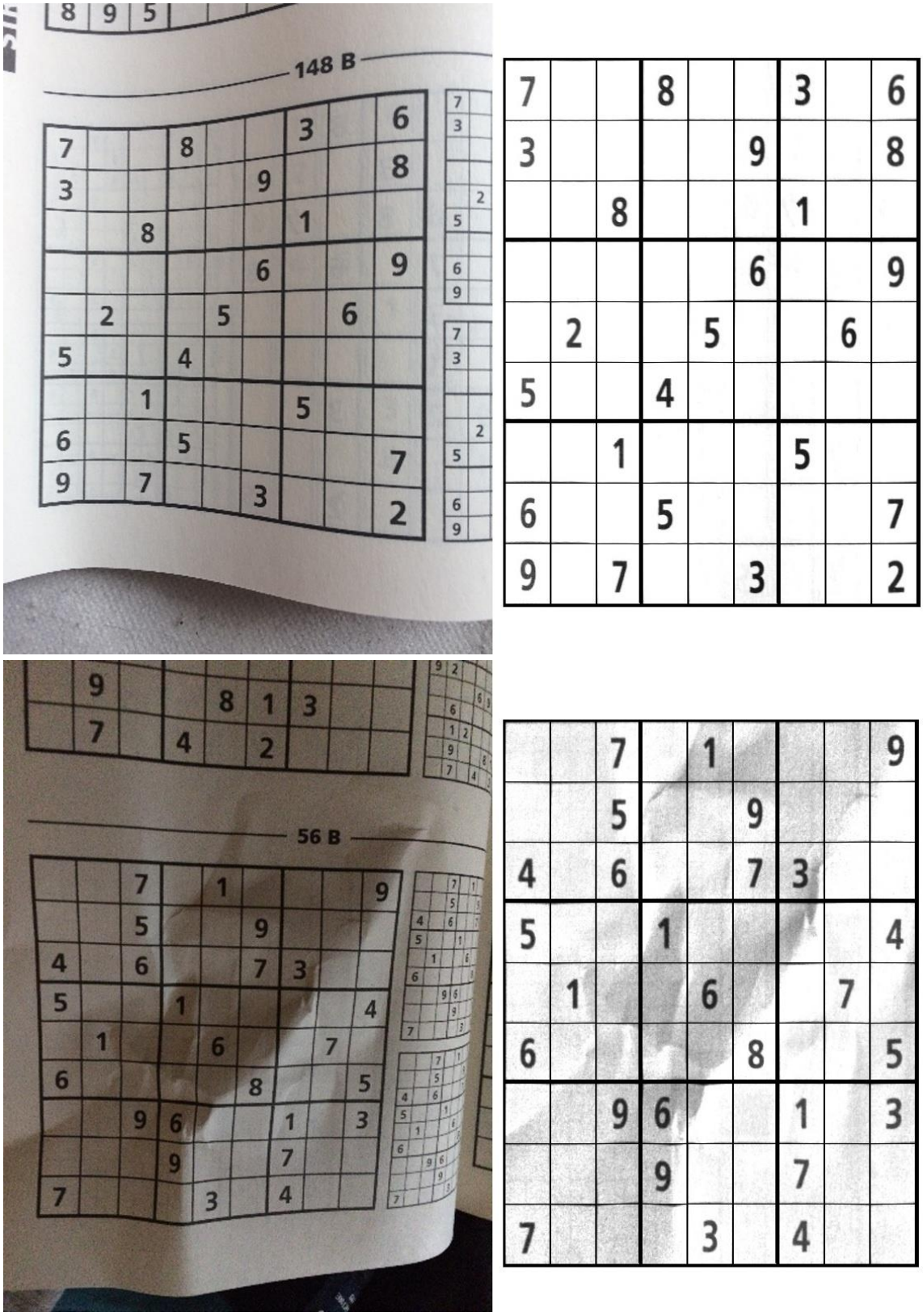
4.2 Zhodnocení výsledků – Detekce sudoku a jiných hlavolamů

Program pro detekci sudoku a jiných hlavolamů využíval pro detekci objektů princip adaptivního prahování. Tento způsob, popsán v kapitole 3.5.3., se osvědčil u této kategorie více, než použití Cannyho detekce hran. Úspěšnost detekce se v tomto případě u testovacích snímků z datasetu blížila k 100%. Při testování Cannyho operátoru se v mnoha případech stávalo, že výsledný obraz hran nebyl u hledané oblasti spojitý. Důvodem byla především tenkost čar, ze kterých byla hledaná mřížka objektu sestavena.

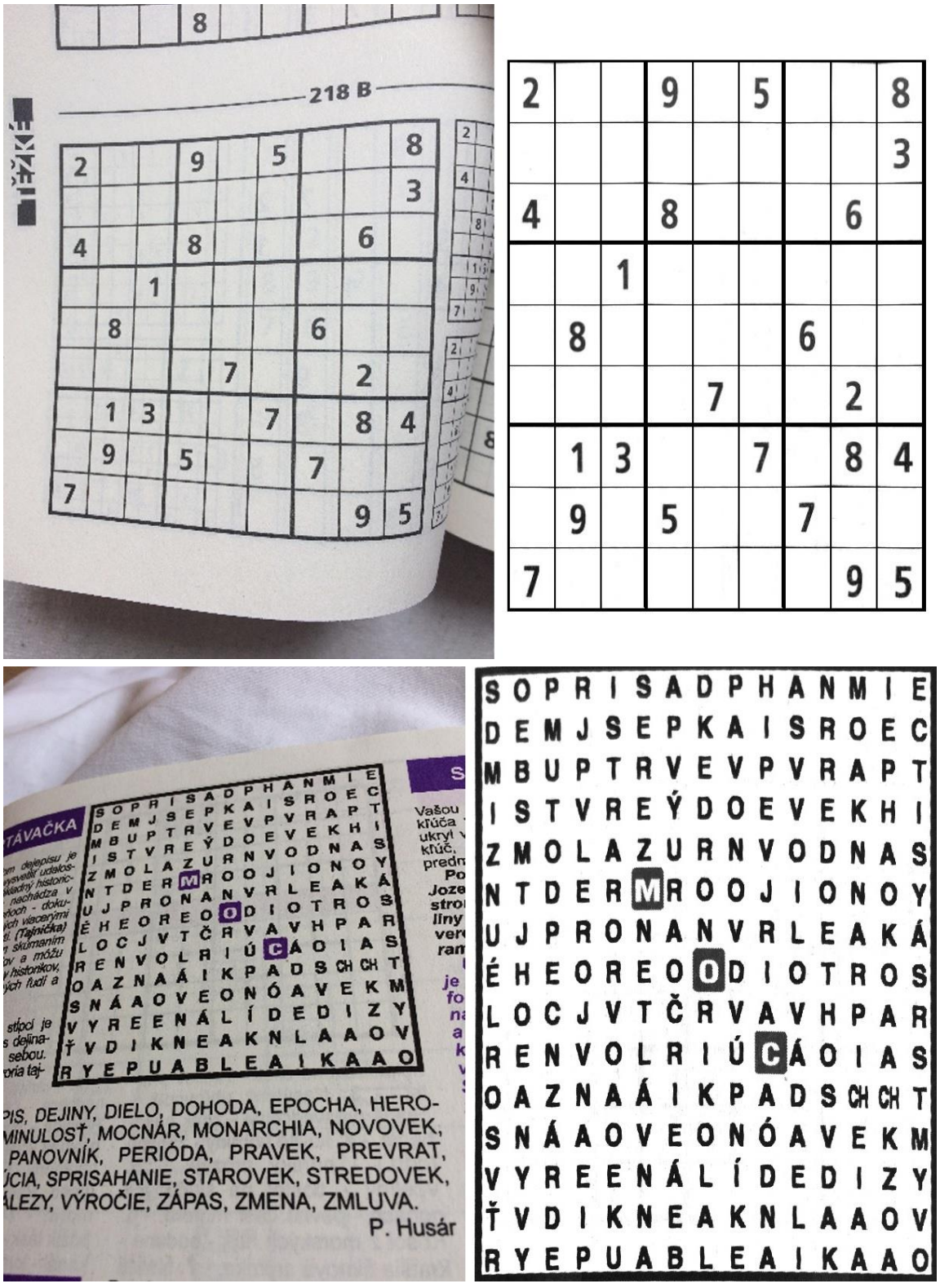
Při provádění opravy perspektivy, program fungoval stejně, jako u kategorie paper_documents bez problémů. Problém představovaly jenom snímky, které trpěly zvlněním papíru. U takových snímků mohlo dojít z důvodu využití jen rohových bodů k ořezání části detekovaného objektu, pokud se jednalo o zvlnění čar směrem ven z oblasti.

Tuto komplikaci program řešil speciálně pro snímky typu sudoku, kde pomocí hledání rohů jednotlivých čtverečků sudoku a následné opravy perspektivy postupně odstraňoval zvlnění papíru. Úspěšnost tohoto algoritmu závisela především na typu vstupních snímků. Proto, aby fungoval, bylo potřeba zvolit vstupní snímky sudoku, kde mřížka nebyla vyfotografována s velkou chybou horizontu. Algoritmus si zároveň nedokázal poradit se snímky s příliš velkým ohnutím horizontálních čar tvořících mřížku. Tyto podmínky vyplývaly ze samotného návrhu algoritmu. Ten využíval jako počáteční bod levý horní roh detekovaného snímku. Následně procházel obrázek v horizontálním směru, kde velikost posunu představovala předpokládaná velikost strany čtyřúhelníku v daném směru. Ta se ale mohla měnit v důsledku zvlnění čar. Zároveň při příliš velkém ohybu čar směrem ven z linie, po které se výsek posouval, pak algoritmus mohl selhat při detekci dílčího čtverečku. Obecně lze říct, že bylo velice obtížné, až nemožné program neparametrizovat pro všechny typy snímků tak, aby se za každých okolností povedlo najít právě hledaný čtvereček. U snímků, které ale splňovaly výše popsané podmínky, program dokázal poměrně kvalitně odstranit problém se zvlněním, jak je možno vidět na obrázcích níže.

Odstraňování stínů a redukce bitové hloubky funguje stejně jako u kategorie č. 1, a pokud se nejednalo o příliš tmavé stíny, dokázal si s nimi program bez problémů poradit.



Obrázek 4-4: Příklady vstupních a výstupních snímků z kategorie sudoku_crosswords



Obrázek 4-5: Příklady vstupních a výstupních snímků z kategorie sudoku_crosswords

4.3 Možnosti vylepšení programů

Jak bylo popsáno v kapitole č. 4, kde proběhlo zhodnocení výsledků, oba programy vykazovaly určité nedostatky, které se projevovaly především v závislosti na vstupních snímcích.

U kategorie paper_documents se jednalo o situace, kde program nedokázal spolehlivě detekovat papír v případě, že pozadí, na kterém se nacházel, nebylo dostatečně kontrastní, nebo naopak bylo příliš rušivé. Tento problém je ale poměrně obtížné řešit a jedním z možných řešení by mohlo být zkoušení různých kombinací morfologických operací a následně hledání spojité kontury. Jinou možností by mohlo být využití Houghovy transformace, která slouží k hledání přímek v obrazu. Pokud bychom byli schopni lokalizovat přímky ohraničující samotný papír a následně najít společné průsečíky, dostali bychom se k souřadnicím všech čtyř rohů papíru.

U kategorie sudoku_crosswords fungovala detekce hledaného objektu velmi spolehlivě, při vyrovnávání mřížky sudoku, by se ale nejspíše také mohla uplatnit Houghova transformace. Pokud bychom dokázali najít všechny přímky tvořící mřížku bez toho, abychom museli využívat výseky obrazu, zbavili bychom se největšího problému, který způsoboval v některých případech nefunkčnost tohoto algoritmu. Tím problémem byla nevhodná volba výseku obrazu, z důvodu nesnadné parametrizace pro všechny možné snímky a následné selhání detekce hledaného čtyřúhelníku.

Oba programy by si zároveň zasloužily vytvoření grafického rozhraní, které sice nemá vliv na funkčnost programu, ale pomáhá zjednodušit práci s programem pro běžného uživatele.

5 ZÁVĚR

V rámci bakalářské práce byly vytvořeny dva programy, které měly za úkol zpracovávat vstupní obrázky, na základě zadání této práce.

V první kapitole práce byl, pro základní pochopení problematiky, vytvořen obecný popis zpracovávání obrazu v oboru počítačového vidění. Následně bylo potřeba určit, jaké typy snímků se budou upravovat.

Pro tuto práci byly vybrány dvě kategorie snímků, kde první kategorii představují snímky papírových dokumentů. Ve druhé kategorii se nachází snímky sudoku a jiných hlavolamů. Cílem bylo navrhnout a vytvořit algoritmy, které budou upravovat snímky z obou kategorii tak, aby výsledné obrázky vypadaly podobně, jako při skutečném skenování papírů.

Ve druhé kapitole proběhla analýza vstupních snímků, která se zaměřovala především na jejich nedostatky a vliv na návrh algoritmů. Pro realizaci těchto algoritmů byl zvolen jazyk python v kombinací s knihovnou OpenCV, která obsahovala mnohé užitečné funkce přínosné pro řešení těchto úkolů.

Samotná realizace, analýza výhod a nevýhod jednotlivých návrhů řešení, byla popsána v kapitole č. 3, která představuje největší část práce.

V kapitole č. 4 je možné vidět zhodnocení výsledků a úspěšnosti jednotlivých programů, včetně názorných ukázek vstupních a výstupních snímků.

Obecně lze říct, že se zadání práce povedlo splnit. Do velké míry to ale záviselo na tom, jaký typ snímků se zadefinoval před vytvářením programů. V problematice zpracovávání obrazů je totiž velmi obtížné naprogramovat algoritmy, které by spolehlivě fungovaly na všechny možné typy snímků.

Přínosem této práce pro mě byla možnost vyzkoušení si programování v novém programovacím jazyku a zároveň možnost zlepšit své dovednosti v oboru zpracovávání obrazů.

BIBLIOGRAFIE

[1] YOUNG, Ian T, Jan J. GERBRANDS a Lucas J VAN VLIET. *Fundamentals Of Image Processing* [online]. Delft, Netherlands: Delft University of Technology, 1998 [cit. 2019-04-25]. ISBN 90–75691–01–7. Dostupné z: https://www.researchgate.net/publication/2890160_Fundamentals_Of_Image_Processing

[2] HLAVÁČ, Václav a Miloš SEDLÁČEK. *Zpracování signálů a obrazů: Pracovní verze skripta v tisku pro studenty*. 7.12.1999. Praha: Vydavatelství ČVUT.

[3] FIŘT, Jaroslav a Radek HOLOTA. *Digitalizace a zpracování obrazu* [online]. In: . Plzeň: Nové technologie – výzkumné centrum, 2015, s. 1-6 [cit. 2019-04-25]. Dostupné z: https://www.researchgate.net/publication/267235327_Digitalizace_a_zpracovani_obrazu

[4] HORAK, Karel, Miloslav RICHTER, Petr PETYOVSKY a Peter HONEC. *Zpracování vícerozměrných signálů: Přednáškové slidy* [online]. In: . Brno: VUT v Brně, 2009-2015 [cit. 2019-04-25]. Dostupné z: http://midas.uamt.feec.vutbr.cz/ZVS/zvs_cz.php

[5] FISHER, Robert, Simon PERKINS, Ashley WALKER a Erik WOLFART. Gaussian Smoothing. *Image processing learning resources* [online]. 2004 [cit. 2019-04-24]. Dostupné z: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

[6] REICHL, Jaroslav a Martin VŠETIČKA. Použité metody převzorkování. *Multimediální Encyklopedie Fyziky*[online]. 2006-2019 [cit. 2019-04-24]. Dostupné z: http://fyzika.jreichl.com/main.article/view/1531-pouzite-metody-prevzorkovani

[7] Grayscale to RGB Conversion. *Tutorials Point: Simply easy learning* [online]. 4th Floor, Incor9 Building, Kavuri Hills, Madhapur, Hyderabad, Telangana 500081 [cit. 2019-04-24]. Dostupné z: https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm

[8] AGU, Emmanuel. *Morphology (Part 2) & Regions in Binary Images (Part 1)* [online]. Worcester, MA | 01609-2280: Computer Science Dept. Worcester Polytechnic Institute (WPI) [cit. 2019-04-24]. Dostupné z: https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture07.pdf

[9] *Morphological Image Processing* [online]. Auckland: The university of Auckland [cit. 2019-04-24]. Dostupné z: https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm

[10] SINHA, Utkarsh. Mathematical Morphology. *AI Shack* [online]. [cit. 2019-04-24]. Dostupné z: http://www.aishack.in/tutorials/mathematical-morphology/

[11] MUTNEJA, Dharampal a Vikram MUTNEJA. *Methods of Image Edge Detection: A Review* [online]. In: . Shaheed Bhagat Singh State Technical Campus, Ferozepur, Punjab, India: Journal of Electrical & Electronic Systems, 2015, 30.07.2015, s. 1-5 [cit. 2019-04-24]. Dostupné z: https://www.omicsonline.org/open-access/methods-of-image-edge-detection-a-review-2332-0796-1000150.php?aid=57249

[12] Canny Edge Detector: Theory. *OpenCV Documentation* [online]. [cit. 2019-04-24]. Dostupné z: https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html?highlight=canny

[13] HLAVÁČ, Václav. *Hledání hran* [online]. Praha: České vysoké učení technické v Praze [cit. 2019-04-24]. Dostupné z: http://people.ciirc.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/22EdgeDetectionCz.pdf

[14] MOKRZYCKI, Wojciech a Marek SAMKO. *New version of Canny edge detection algorithm* [online]. In: . Olsztyn, Poland: Faculty of Mathematics and Informatics, University of Warmia and Mazury, 2012, Leden 2012, s. 533-540 [cit. 2019-04-24]. Dostupné z: https://www.researchgate.net/publication/236024099_New_version_of_Canny_edge_detection_algorithm

[15] SINHA, Utkarsh. Fundamentals of Features and Corners: Harris Corner Detector. *AI Shack* [online]. [cit. 2019-04-24]. Dostupné z: http://aishack.in/tutorials/harris-corner-detector/

[16] Contour Features. *OpenCV Documentation* [online]. [cit. 2019-04-24]. Dostupné z: https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

[17] *Finding Eigenvalues and eigenvectors* [online]. In: . Dublin: Trinity College Dublin, s. 1-4 [cit. 2019-04-24]. Dostupné z: https://www.scss.tcd.ie/~dahyotr/CS1BA1/SolutionEigen.pdf

[18] COLLINS, Robert. *Harris Corner Detector* [online]. In: . University Park, PA 16802, USA: PennState College of Engineering, s. 1-27 [cit. 2019-04-25]. Dostupné z: http://www.cse.psu.edu/~rtc12/CSE486/lecture06.pdf

[19] SINHA, Utkarsh. Fundamentals of Features and Corners: The Shi-Tomasi Corner Detector. *AI Shack* [online]. [cit. 2019-04-24]. Dostupné z: http://aishack.in/tutorials/shitomasi-corner-detector/

[20] Perspektiva ve fotografii. *Fotografování* [online]. 27.02.2014 [cit. 2019-04-24]. Dostupné z: http://www.fotografovani.cz/fotopraxe/zakladni-postupy1/perspektiva-ve-fotografii-jak-na-ni--165970cz

[21] DE VRIES, Jelmer Philip. *Object Recognition: A Shape-Based Approach using Artificial Neural Networks*[online]. In: . Utrecht: Utrecht University, Leden 2006 [cit. 2019-04-24]. Dostupné z: https://www.researchgate.net/publication/251509419_Object_Recognition_A_Shape-Based_Approach_using_Artificial_Neural_Networks

[22] FUSSELL, Don. *Affine Transformations* [online]. In: . Austin,USA: University of Texas at Austin, 2010, 2010, s. 1-33 [cit. 2019-04-25]. Dostupné z: https://www.cs.utexas.edu/users/fussell/courses/cs384g-fall2011/lectures/lecture07-Affine.pdf

[23] HOUSE, Donald H. *Affine Transformations* [online]. In: . Clemson, S.C, USA: The School of Computing at Clemson [cit. 2019-04-25]. Dostupné z: https://people.cs.clemson.edu/~dhouse/courses/401/notes/affines-matrices.pdf

[24] HECKBERT, Paul. *Fundamentals of Texture Mapping and Image Warping: Projective Mappings for Image Warping* [online]. In: . Berkeley, CA, USA: U.C. Berkeley, 1999, 13.09.1999, s. 1-5 [cit. 2019-04-25]. Dostupné z: http://graphics.cs.cmu.edu/courses/15-463/2008_fall/Papers/proj.pdf

[25] Affine and Projective Transformations. *Graphics Mill* [online]. 901 N. Pitt Street, Suite 325 Alexandria, VA 22314, United States: Aurigma [cit. 2019-04-25]. Dostupné z: https://www.graphicsmill.com/docs/gm5/Transformations.htm

[26] DRAKOS, Nikos a Ross MOORE. *Histogram and normalization* [online]. In: . Computer Based Learning Unit, University of Leeds. Mathematics Department, Macquarie University, Sydney [cit. 2019-04-25]. Dostupné z: http://fourier.eng.hmc.edu/e161/lectures/digital_image/node9.html

[27] Miscellaneous Image Transformations: threshold. *OpenCV Documentation* [online]. [cit. 2019-04-25]. Dostupné z: https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html?highlight=threshold#cv.Threshold

[28] Image Thresholding. *OpenCV Documentation* [online]. [cit. 2019-04-25]. Dostupné z: https://docs.opencv.org/3.2.0/d7/d4d/tutorial_py_thresholding.html

[29] SRIRAM. Sudoku Solver - Real time Image. *Sriram's Blog* [online]. [cit. 2019-04-25]. Dostupné z: http://nnsriram.blogspot.com/2015/12/sudoku-solver-real-time-image.html

[30] LAI, Andrew. A computer vision based sudoku solver: Built on OpenCV and Python. *Returnvector* [online]. 2015 [cit. 2019-04-25]. Dostupné z: https://returnvector.net/sudoku.html

[31] SÝKORA, Petr. *Velká kniha sudoku*. Praha: PLOT, 2014.

[32] *Relax: krížovkársky dvojtýždenník*. Banská Bystrica: TRIAN, 2006, **XVII.**(15/06).

[32] *Relax: krížovkársky dvojtýždenník*. Banská Bystrica: TRIAN, 2018, **XXIX.**(17/18).

SEZNAM PŘÍLOH

Příloha této bakalářské práce uložená na CD nosiči obsahuje následující složky:

1. Složku **paper_documents**, ve které se nachází program se stejnojmenným názvem paper_documents.py, 30 snímků papírových dokumentů a několik obecných snímků

2. Složku **sudoku_crosswords**, ve které se nachází program se stejnojmenným názvem sudoku_crosswords.py, 30 snímků typu sudoku a 20 snímků typu křížovky nebo jiné hlavolamy

3. Složku **bakalářská_práce_xpiege00**, která obsahuje text práce ve formátu pdf