VŠB - TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

# BAKALÁŘSKÁ PRÁCE

2012 Tomáš Straka

VŠB - TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
KATEDRA INFORMATIKY

# Kopírování dokumentů pomocí kamery

# Scanning Documents Using Camera

2012 Tomáš Straka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání bakalářské práce

Student: **Tomáš Straka**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Kopírování dokumentů pomocí kamery
Scanning Documents Using Camera

Zásady pro vypracování:

V některých případech je nutné rychle připravit kopii dokumentu bez použití skeneru. Digitální fotoaparáty, mobilní telefony a webové kamery jsou dnes již velice běžné zařízení a není problém pořídit fotografii v jakékoliv situaci. Úkolem této práce je vytvoření aplikace, která umožní získat kopii dokumentu vyfoceného fotoaparátem nebo kamerou. Tato aplikace by měla být schopná najít oblast snímaného dokumentu a pomocí geometrických transformací a obrazových filtrů připravit snímaný dokument k zobrazení na obrazovce nebo k tisku.

1. Nastudovat existující metody.
2. Vytvořit funkční implementaci.
3. Připravit sadu vhodných obrazových filtrů.
3. Kriticky zhodnotit svou implementaci.

Seznam doporučené odborné literatury:

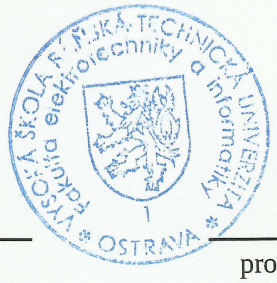
Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Karel Mozdřeň**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012




doc. Dr. Ing. Eduard Sojka
*vedoucí katedry*

prof. RNDr. Václav Snášel, CSc.
*děkan fakulty*

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 4.5.2012 Podpis: .....


## Poděkování

Rád bych poděkoval Ing. Karlu Mozdřeňovi za cenné rady, které mi pomohly při tvorbě mé bakalářské práce. Dále za konzultace, u kterých mi poskytl mnoho užitečných informací a podkladů k řešení.

## Abstrakt

Tato bakalářská práce se zabývá kopírováním textu pomocí fotoaparátu a webových kamer. V úvodní části vysvětluje současný stav, možnosti kopírování textu a popisuje základní používané postupy. Následující část práce je zaměřena na teoretické principy, které jsou rozebrány a sepsány. Tyto principy jsou uplatněny v praktické části se samotným řešením problému. Nakonec jsou provedeny experimenty ověřující správnost řešení problematiky a tyto experimenty jsou doplněny o praktické ukázky. V závěru je shrnuta řešená práce a jsou zhodnoceny výsledky.



**Klíčová slova:** *kopírování dokumentů, zpracování obrazu, transformace obrazu, fotoaparát, OCR*



## Abstract

This bachelor thesis deals with text copying using the camera and webcamera. The introduction section explains current situation, possibilities of text copiing and describes the basic procedures used. Next section of the work is focused on theoretical principles, that are analyzed and described. These principles are applied in the practical part with the solution of the problem. Verifivation of the solution correctness is demonstrated using practical aplications. The conclusion summarizes this thesis and evaluates results.



**Keywords:** *copying documents, image processing, image transformation, camera, OCR*

# Obsah

**1 Úvod** **2**

**2 State of The Art** **3**
2.1 Snímání pomocí fotoaparátu . . . . . 4

**3 Návrh řešení** **5**
3.1 Transformace . . . . . 5
3.1.1 Výřez . . . . . 6
3.1.2 Rotace . . . . . 7
3.1.3 Změna velikosti . . . . . 9
3.1.4 Prostorová transformace . . . . . 10
3.2 Filtry . . . . . 11
3.2.1 Stupně šedi . . . . . 11
3.2.2 Prahování . . . . . 12
3.2.3 Jas a kontrast . . . . . 13
3.2.4 Potlačení šumu . . . . . 14
3.3 OCR . . . . . 15

**4 Řešení** **16**
4.1 Úvodní okno programu . . . . . 16
4.2 Získání obrazu z webkamery . . . . . 17
4.3 Úprava obrazu . . . . . 20
4.4 Nápověda k programu . . . . . 24
4.5 Výstupní obrázky . . . . . 25

**5 Experimenty** **26**
5.1 Časové testy . . . . . 26
5.2 Úspěšnostní testy . . . . . 27

**6 Závěr** **30**

# 1 Úvod

V dnešní době existuje mnoho zařízení pro kopírování dokumentů, například kopírky a skenery. Některá zařízení umí kopírovat dokumenty přestože, k tomu nejsou primárně určena, například fotoaparáty. Zařízení ke kopírování dokumentů jsou vyvinuta jednotlivými výrobci tak, aby jejich kopie byly co nejkvalitnější a zároveň reálně zobrazovaly předlohu. Dalším aspektem v tomto ohledu je určitě čas, kdy zařízení by mělo za co nejkratší dobu zvládnout co nejvíce kopií. V neposlední řadě se od zařízení požaduje co možná nejsnadnější instalace a obsluha. Za standardní funkci na kopírkách se považuje takzvané vytvoření kopie jedním stiskem tlačítka.

Nejčastěji se pomocí kopírek a skenerů kopírují textové dokumenty, u nichž je rozhodujícím faktorem rychlost vytvoření kopie z originálního dokumentu. Jelikož tyto dokumenty obsahují nejčastěji pouze kontrastní černý text na bílém papíře, jsou kopie vytvářeny velmi rychle a zařízení nepotřebuje k nasnímání tak vysoké rozlišení jako v případě fotografií. Rozlišení u elektronických dokumentů se udává v jednotkách DPI (počet bodů na palec).

Občas se můžeme doslechnout či dočíst z médií, že byly zkopírovány staré nebo archivní mapy, atlasy či jiné vzácné dokumenty, u nichž můžeme říci, že se jedná o digitalizaci dokumentů. Jelikož se často jedná o historicky cenné unikáty a mnohdy i jediné dochované kusy, najímají muzea a jiné ústavy specializované firmy k digitalizaci takových dokumentů. Tyto firmy používají speciální nedestruktivní metody a „namíru„ upravené zařízení ke kopírování. Protože se často jedná o velmi drahá díla kde sebemenší chyba může mít nedozírné následky.

Ve své bakalářské práci se zabývám metodami kopírování dokumentů. Snažím se najít postupy, abych dosáhl co možná nejlepších výsledků. Vytvořím jednoduchý program, který bude umožňovat zachycení dokumentu pomocí libovolné webkamery. Na tento zachycený dokument se budu snažit vybrat a použít jednotlivé filtry, umožňující oddělit text dokumentu od okolního prostředí a od papíru, na kterém je napsán. Upravený dokument bude nutné transformovat geometrickými transformacemi tak, aby co nejvíce vystihoval předlohu, ze které byl pořízen. Jednotlivými vhodnými transformacemi a filtry na úpravu textu se budu zabývat v kapitole návrhu řešení.

Hlavním cílem mé práce je vytvoření programu na úpravu nasnímaného textu. Program umožní upravit zachycený dokument tak, aby byl čitelnější, než tomu bylo u dokumentu před úpravami. Výsledek by měl splňovat podmínku lepší čitelnosti pro překladače textu z rastrové podoby. Dále bude kladen důraz na následné možnosti práce s upraveným dokumentem, tudíž nesmí chybět možnost dokument uložit nebo jej přímo vytisknout na tiskárně.

V závěru mé práce jsem provedl experimenty a naplánoval jednoduché testy. Ty mají mají za úkol ověřit funkčnost vytvořeného programu. Jedná se o testy k porovnání rychlosti a kvality vybraných algoritmů pro úpravu textu. Využitím vlastního programu pro testování ověřuji účinnost algoritmů. Všechny výsledky jsou doplněny obrázky, na kterých je text jak originální, tak i v upravené podobě. Vlastní výsledky algoritmů jsou porovnány.

# 2 State of The Art

Existují zařízení k zachytávání obrazu či videosekvencí, která nejsou primárně určena ke kopírování. Těmito zařízeními nejčastěji zachytáváme obrazy z reálného světa, ať už jsou to lidé, rostliny nebo třeba automobily. Lze s jejich pomocí také kopírovat textové či grafické dokumenty, avšak s jistým omezením. Vezměme v úvahu například digitální fotoaprát. U většiny starších a u všech nových modelů můžeme nastavit speciální režimy, které nám dovolují vyfotit předměty zblízka. Tyto režimy se nazývají makro a supermakro. S režimy makro, respektive supermakro, můžeme zaostřit dokument již od několika málo centimetrů. Avšak i přes použití těchto režimů nám nastávají další problémy například s citlivostí snímače nebo s nasvícením dokumentu.

Existují také samozřejmě zařízení speciálně zkonstruovaná pro kopírování dokumentů. Můžeme se setkat s příručními zařízeními, které člověk uchopí do ruky a s nimiž přejíždí po papíře a tímto způsobem nasnímá požadovaný dokument. Toto zařízení pak jen stačí připojit k počítači a zpracovat nasnímané fotografie. Jednou z dalších možností, kterou současný trh nabízí, jsou jednoduché, lampičkám podobné produkty (Obr. 1), které jsou takto uzpůsobeny z důvodů jednoduché manipulace a lehké přenositelnosti. Samotné zařízení obsahuje vlastní zdroj světla a vestavěnou kameru. Zařízení je konstruované tak, že obsahuje kontrastní podložku, na níž jsou zvýrazněna místa, na která se položí různé velikosti dokumentů. Přístroj je přesně nastaven, aby na vyznačených místech ideálně nasnímal požadovaný dokument.

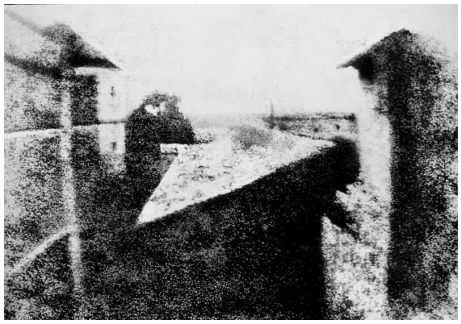
Obrázek 1: Přístroj specializovaný pro kopírování textu

K těmto veřejnosti dostupným nástrojům na kopírování textu ještě existují specializované přístroje, které umožňují nasnímání dokumentů. Tyto stroje jsou především určené pro kopírování knih o několika stovkách či tisících stran a jejich konstrukce umožňuje prosté vložení a nastavení knihy do správné polohy, přičemž není nutné knihy po každém nasnímání stránky otáčet, ale stroj se stará o otáčení listů knih sám.

## 2.1 Snímání pomocí fotoaparátu

Představme si situaci, že nemáme k dispozici kopírku ani skener a potřebujeme nasnímat dokument v papírové podobě. V takovém případě se musíme poohlédnout po něčem, co by zvládlo tento úkol. Nejčastěji můžeme použít digitální fotoaparát nebo mobilní telefon, který jej má zabudovaný. V dnešní době, kdy ve světě vládne hon na výrobek, jenž má co největší rozlišení, by neměl být problém kvalitně zachytit dokument. Avšak počet megapixelů ještě z přístroje nedělá zařízení, které dokáže kvalitně zachytit daný dokument. Jsou zde i další aspekty ovlivňující výslednou kvalitu fotografie.

Už první fotoaparáty fungovaly na principu zachycení světelného odrazu, avšak světlo se nezachytávalo na světlocitlivý čip, který se používá v dnešní době nebo na světlocitlivou pásku používanou v minulém století. Ale princip byl obdobný jako je tomu dnes. Autorem nejstarší fotografie je Francouz Joseph Nicéphore Niepce, kterému se po několikahodinovém čekání podařila vytvořit první fotografie, kterou nazval Pohled z okna (Obr. 2). Fotografie byla vytvořena na cínové destičce pokryté roztokem.



Obrázek 2: Joseph Nicéphore Niepce: Pohled z okna

Kvalitní světlo umožňuje zachytit snímek téměř bez šumu. Nejvhodnějším světlem je světlo sluneční. Nevhodná jsou světla příliš ostrá a intenzivní. Následkem ostrého světla mohou vznikat na výsledném snímku odlesky a přepaly, se kterými si většinou optika neumí příliš poradit. Ideální ve vnitřních prostorách je použít bílé světlo rozmístěné okolo dokumentu, který chceme zachytit. Minimálně ze dvou, ale nejlépe ze tří stran tvořících trojúhelník. Nejlepší fotografie jsem získal při sevření úhlu fotoaparátu vůči podložce 45 stupňů.

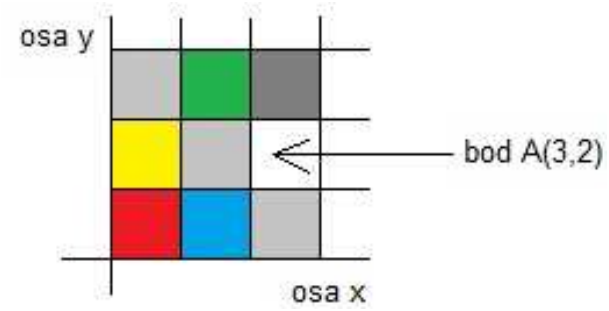
Užitečnou funkcí, se kterou se setkáváme u novějších typů fotoaparátů, je stabilizace obrazu. Existuje několik variant stabilizací, s fotoaparáty je nejčastěji spojena digitální stabilizace obrazu. Jedná se o softwarovou funkci fotoaparátu, která umí snímek částečně stabilizovat pomocí posouvání okrajů obrazu mimo objektiv. Software pak jen dopočítává stejné body na obraze a jejich posunutí. Dále existují metody, které stabilizují obraz až po záznamu.

Při fotografování dokumentu je dobré použít stativ, který bude přístroj fixovat na jednom místě a odstraní se tak drobné nebo větší otřesy rukou, které se mohou jevit zanedbatelné. Ačkoliv se nám může zdát, že přístroj držíme v rukou bez hnutí, opak je pravdou. Drobné otřesy, způsobené cukáním svalů nebo srdečním tepem ani nepostřehneme. Navíc fotoaparát neudržíme v rukou v klidu na jednom místě delší dobu. Pokud přece jen stativ nemáme, je dobré použít předměty v okolí, které vyhovují našemu kritériu podložení fotoaparátu.

# 3 Návrh řešení

Zachycený dokument ve formě fotografie není dokonalý a vyžaduje úpravy, aby se dal s co nejmenšími rozdíly od originálního dokumentu vytisknout či uchovat v elektronické podobě. Po analýze několika vyfotografovaných dokumentů jsem došel k závěru, že by bylo potřeba některé dokumenty více či méně upravit. Pro úpravu je dobré využít geometrických transformací, ať už základních, někdy uváděných jako afinní transformace [4], tak i složitějších transformací v prostoru, které jsou v naprosté většině složeny ze základních transformací.

Další důležitou úpravou je odfiltrování nežádoucích jevů na fotografii, k tomuto postupu je vhodné použít ověřené filtry a postupy, které tomu napomohou. V následující části budu pracovat s rastrovou grafikou, neboli s obrázky či objekty, které jsou tvořeny dvourozměrnou sítí bodů (Obr. 3), kde tato síť je definovaná vodorovnou osou $x$ a svislou osou $y$. Každý z bodů je definován svojí polohou v síti a mimo polohy ještě svými vlastnostmi (jasem, barvou...).




Obrázek 3: Rastrový obrázek, bod se souřadnicemi (x,y)

## 3.1 Transformace

Mezi základní geometrické transformace [4][5] se řadí posunutí, otáčení, změna měřítka a zkosení, avšak ne všechny zde jmenované jsou pro úpravu fotografií vhodné. Například v posunutí textu nevidím velký smysl a na místo něj bych použil vyříznutí obrázku. Transformace je vlastně jen zobrazení bodu ze zdrojového obrazu na bod obrazu cílového, přičemž se mění jeho poloha v obraze podle pravidel použité transformace, neboli se jedná o transformovaný obraz zdrojového bodu s jinými či stejnými souřadnicemi.

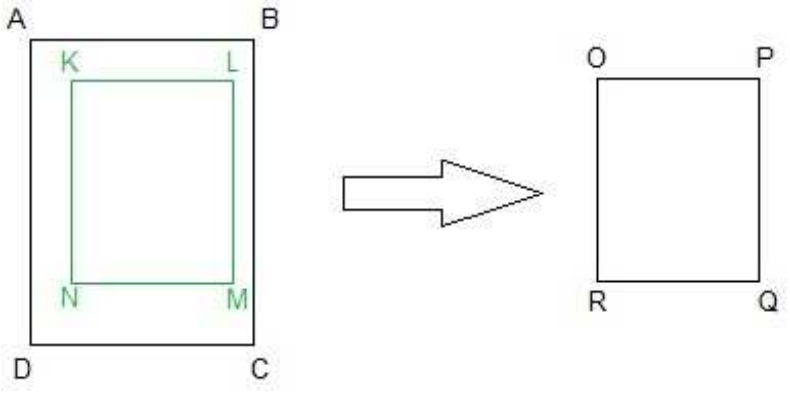
Například: Mám bod $a$ v obraze se souřadnicemi $x, y$ reprezentujícími souřadnice bodu. Po použití transformace $T$ vzniká na výsledném obraze bod $a$‘ se souřadnicemi $x$‘, $y$‘. Transformace jsou nejčastěji reprezentovány transformačními maticemi, zdrojové souřadnice bodu jsou maticí vynásobeny a tím vznikají výsledné souřadnice obrazu bodu. Pro použití transformací je vhodné použití Homogenního souřadnicového systému namísto Kartézského. Hlavní výhodou je možnost použití všech transformací jako matic a jejich jednoduché skládání.

$$a‘ = T(a) \tag{1}$$

Schéma použití trasformace $T$ na bod $a$

### 3.1.1 Výřez

U zachyceného obrazu textu se na okrajích stránek mohou vyskytovat prvky, které si nepřeji zkopírovat, mohou to být různé popisky, značky či třeba části prstů, kterými jsem držel stránku. S dokumentem může být zachycen také okolní prostor, který není vhodný kopírovat. Jednoduchým řešením jak se nežádoucích prvků zbavit je použití výřezu dokumentu, v mém případě obrázku dokumentu. Vytvoření výřezu jednoduše provedeme tak, že vytvoříme nový obrázek, jehož velikost a obsah se bude rovnat velikosti a obsahu výřezu původního obrázku (Obr. 4).




Obrázek 4: Schéma výřezu obrázku

Mám zdrojový obrázek, který je reprezentován ve schématu obdélníkem $ABCD$ a chci vytvořit výřez, který je reprezentován obdélníkem $KLMN$. K vytvoření výběru potřebuji znát rohový bod původního obrázku, který je považován za počátek. U počítačového zobrazení se za počátek nejčastěji považuje levý horní roh, což je v mém případě bod $A$. K tomuto bodu mi stačí znát protilehlé souřadnice bodu, v mém případě bod $C$, souřadnice zbylých dvou bodů lze jednoduše dopočítat. Nyní mi zbývá získat souřadnice dvou protilehlých bodů výběru, například bodu $K$ a $M$. Všechny potřebné údaje již znám a stačí mi vytvořit nový obdélník, jehož velikost se bude rovnat velikosti obdélníku $KLMN$ a jeho body budou odpovídat výběru.

Početně: mám bod $A(0,0)$, bod $C(150,110)$, bod $K(20,20)$, $M(100,120)$ potřebuji vypočítat velikost obdelníku $OPQR$, neboli souřadnice bodu $Q$:

$$Qx = (Mx - Kx) = (100 - 20) = 80 \tag{2}$$
$$Qy = (My - Ky) = (120 - 20) = 100. \tag{3}$$

Výpočet souřadnic bodu $Q$

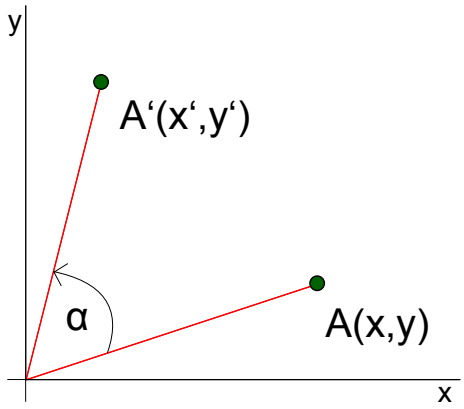
Nyní můžu vytvořit nový obrázek, jehož velikost je 80×100 bodů. Zbývá už jen rekurzivně naplnit nový obrázek body, přičemž bod $V$‘ v obdélníku $OPQR$ bude odpovídat bodu $V$ v původním výběru $KLMN$.

$$V‘(x‘, y‘) = (V(x, y) + K(x, y)) \tag{4}$$

Vztah obrazu bodu $V$‘ a bodu $V$, kde (x,y) jsou souřadnice bodu

### 3.1.2 Rotace

Mezi standardní operace transformace obrazu patří rotace [4][5], neboli otočení obrazu. Rotace obrazu umožňuje otočit jakýkoli obrázek o libovolný kladný či záporný úhel a lze ji provádět v dvourozměrném (2D) či trojrozměrném (3D) prostoru a to vůči libovolné ose či bodu. Pro mé potřeby budu používat rotaci obrázku v dvourozměrném prostoru a vůči jednomu bodu. Rotovaný obraz může zasahovat mimo vykreslovaný prostor. Proto je nutné tuto vlastnost ošetřit změnou velikosti obrazu, jinak dojde k jeho oříznutí. Nejjednodušší je před samotným otočením si spočítat výsledné souřadnice rohových bodů, vybrat maximální $x$ a $y$ souřadnici a podle souřadnic upravit obraz. U rotace podle počátku bychom měli mít na paměti, že počátek v počítačové grafice se nenachází vlevo dole, ale standardně vlevo nahoře a s tím také počítat při práci s rotací.




Obrázek 5: Rotace bodu o úhel alfa

K výpočtu rotace kolem počátku mi stačí znát velikost úhlu, o který budu objekt otáčet. Vzorec (5)(6) je rozdělen na dvě části, v nichž se počítají výsledné hodnoty $x‘$ a $y‘$ ze zadaného bodu, který má polohu $x$ a $y$. Ve vzorci je použito standardní označení úhlu a to řecké písmeno alfa ($\alpha$).

$$x‘ = x * \cos(\alpha) - y * \sin(\alpha) \tag{5}$$
$$y‘ = x * \sin(\alpha) + y * \cos(\alpha) \tag{6}$$

Vzorec pro rotaci kolem počátku.

Pro výpočet rotace kolem obecného bodu musím znát mimo velikosti úhlu, o který chci objekt otočit, také polohu bodu, podle kterého budu otáčet. Nejčastěji se volí středový bod obrázku, který lze jednoduše získat dělením: $Ax$ = šířka/2 , $Ay$ = výška/2. Výsledkem je bod ve středu obrázku, kolem něhož se bude celý obrázek otáčet. Lze však samozřejmě zvolit libovolný bod, kolem kterého se bude obrázek otáčet.

$$x‘ = Ax + (x - Ax) * \cos(\alpha) - (y - Ay) * \sin(\alpha) \tag{7}$$
$$y‘ = Ay + (x - Ax) * \sin(\alpha) + (y - Ay) * \cos(\alpha) \tag{8}$$

Vzorec pro rotaci kolem obecného bodu.

Při otáčení obrázku vznikají prázdné body, které jsou způsobeny převáděním výsledků rotace na celočíselný formát. Už při otočení o pouhý jeden stupeň se začínají na výsledném obraze tyto body objevovat, avšak při otočení o 90°, 180°, 270° a 360° se logicky tyto body nevyskytují. Protože dochází k celočíselným výsledkům, zachovává se jak délka, tak šířka obrazu. Obraz se pouze převrací podle zvoleného úhlu. K redukci těchto prázdných bodů se používá metoda interpolace [7] a jedná se o metodu, která umožňuje podle vzorce vyplnit prázdné body. V současné době existuje mnoho druhů interpolací, lišících se především rychlostí a kvalitou, přičemž pokud budeme chtít zvýšit kvalitu interpolace, musíme počítat s tím, že se zvýší časová náročnost výpočtu.

Nejrychlejší metodou interpolace je metoda nazývána nejbližší soused. Tato metoda k vyplnění bodu vybere bod, který je vybranému bodu nejblíže a pouze zkopíruje jeho vlastnosti. Složitější interpolací je takzvaná bilineární interpolace, která se skládá ze dvou interpolací, interpolací ze směru $x$, tak i interpolací ze směru $y$. Postupem bilineární interpolace je použít interpolaci na 4 rohové body okolo vybraného bodu. Výpočetně náročnější jsou interpolace bikubická, spline nebo sinc. Prvním obrázkem ze všech 4 (Obr. 6) je obrázek původní, všechny následující byly otočeny o 20°. Na druhý obrázek nebyla použita žádná interpolace a vyskytuje se na něm mnoho prázdných bodů znázorněných bílými body. Na třetí obrázek byla použita interpolace nejbližší soused a lze si všimnout, že obrázek je značně kostrbatý. Na posledním obrázku byla použita bilineární interpolace, jenž má lepší výsledky na kvalitě oproti předchozí metodě, avšak je časově náročnější.

Obrázek 6: Originální obrázek, bez interpolace, interpolace nejbližší soused, bilineární interpolace

### 3.1.3 Změna velikosti

Za změnu velikosti obrazu [4] se považuje zmenšení či zvětšení obrazu, neboli použití takzvaného zoomu udávaného v procentech, nebo v násobcích - např: zoom 200 % = zoom 2x. Dvojnásobný zoom zvětší obraz na délce i výšce dvakrát, přičemž počet bodů obrazu se zčtyřnásobí. Při rozdílném zmenšování či zvětšování šířky obrazu od výšky, dochází k deformaci obrazu a to natáhnutí v ose $x$ či ose $y$. Při zvětšování obrazu nastává problém prázdných bodů v obraze, podobně jako tomu bylo u otáčení. Při zvětšování vznikají v obraze prázdná místa (Obr. 6).

U zvětšení na 200 % je každý druhý řádek i sloupec tvořen prázdnými body. Stejně jako u otáčení se na zaplnění bodů používá interpolace. První obrázek je originální, ostatní obrázky jsou zvětšeny o 10 % (Obr. 7). Druhý obrázek je bez použití interpolace a prázdné body jsou zde znázorněny bílou mřížkou. Po deseti bodech obrazu následuje vždy prázdný bod jak na ose $x$, tak $y$. U třetího obrázku byla použita interpolace nejbližší soused a u posledního, neboli čtvrtého, interpolace bilineární.

Obrázek 7: Originální obrázek, bez interpolace, interpolace nejbližší soused, bilineární interpolace

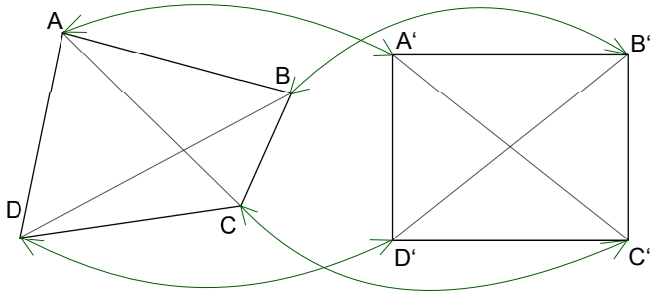
Protikladem zvětšení je zmenšení fotografie, kdy se místo přidávání naopak body ubírají. Problém je, které body by se měly ubrat. Obraz potřebujeme zmenšit na polovinu, jak ke zmenšení přistupovat? Existuje několik metod jak řešit zmenšení fotografií. První z nich je násilné zmenšení obrazu, kdy naráz dojde ke zmenšení obrazu o celou polovinu v obou směrech. Toto řešení není příliš šťastné, protože můžeme přijít o podstatná data v obraze. Druhou metodou, která se nejvíce využívá, je metoda postupného zmenšování, kdy se obraz zmenšuje v několika krocích. Za novou metodu zmenšování, by se dala považovat metoda seam carving [8], která před zmenšením obraz analyzuje a označí si místa hodnotami. Vysokou hodnotou, pokud jsou místa kontrastní a hodně detailní nebo naopak místům, jako jsou plochy, přiřazuje hodnoty nízké (může se jednat o vodní hladinu, oblohu...). Poté provede samotné ořezávání, kdy střídavě z šířky a výšky odkrajuje body z obrazu s nejnižšími hodnotami.

### 3.1.4 Prostorová transformace

Pokud budu chtít zachytit papír s textem, většinou se mi nepodaří zachytit obraz přesně kolmo nad papírem a vyfotografovaný text bude na výsledném obrázku částečně zapuštěn do prostoru. Nejjednodušším způsobem, jak z tohoto obrázku dostat papír s textem, je transformovat obraz z prostoru.

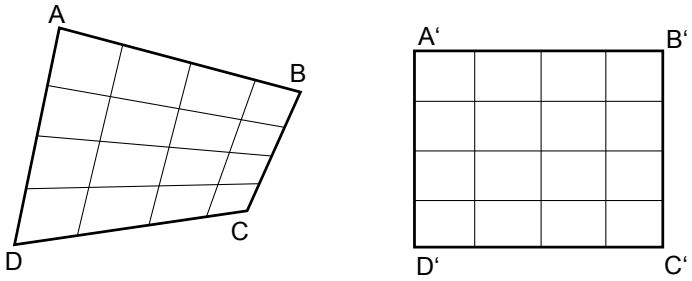
Jednou z nejpoužívanějších a nejrozšířenějších metod je homografie [10], která umožňuje mapovat bod obrazu na bod obrazu jiného. Avšak já budu pracovat pouze s čtyřúhelníkovými objekty, proto je vhodnější využít metodu transformaci čtyřúhelníku (Quadrilateral transformation) [9]. Základem pro použití této transformace je nutnost znát všechny čtyři rohové body papíru v obrazu a cílové body čtvercového obrazu. U této transformace se můžeme setkat s projektivní metodou (Obr. 8), afinní transformací nebo třeba s polynomickou transformací.

U projektivní metody se namapují jednotlivé souřadnice původního čtyřúhelníku na výsledný obdélník, to znamená, že jednotlivé body a jejich obrazy si budou odpovídat, například: bod $A(x, y) = A'(x', y')$. Ze zdrojových i cílových bodů se vytvoří transformační matice, která se poté aplikuje na celý obraz, který chceme převést. Pro výpočet výsledných koordinátů se zdrojové souřadnice bodu převádí do homogenní soustavy, kde se rozšíří o jeden prostor označený symbolem w, braný jako váha ($A(x, y) \rightarrow A(x, y, w)$). Tím se usnadní následný výpočet, protože poté lze jednoduše vynásobit souřadnice bodů s transformační maticí. Na výsledný obrázek bude nutné použít interpolaci.




Obrázek 8: Projekce čtyřúhelníku

Jedním ze způsobů, jak řešit problém, je rozdělení zdrojového obrázku na menší části (Obr.9), přičemž každá strana obrazu se rozdělí na $n$ částí, následně pospojuje a bude vytvářet jakousi síť, kde jednotlivé vzniklé menší čtyřúhelníky budou odpovídat čtyřúhelníkům ve výsledném obdélníku, který je rozdělen stejným způsobem.




Obrázek 9: Rozdělení obrazu na menší části

## 3.2 Filtry

Pro další úpravu obrázku je dobré využít některých filtrů. Obrazové filtry umožňují upravit zdrojový obrázek tak, že se zachovává velikost a tvar obrazu a mění se jeho obsah. Nejčastěji se využívají filtry, které upravují barevnost, intenzitu a ostrost obrazu. Tyto filtry mohou vylepšit vlastnosti obrázku a odfiltrovat vlivy způsobené zachycením dokumentu.

Nejpoužívanější barevný model [1] v počítačové grafice je model RGB, u kterého se pro výslednou barvu bodu míchají barvy červená (R - red), zelená (G - green) a modrá (B - blue) v různých intenzitách. Existuje však mnoho dalších modelů, které se běžně používají, ať je to barevný model CMY(K) pro tisk nebo HSL v počítačové grafice a další. Pro model RGB existuje rozmanitá škála intenzit, takzvaná barevná hloubka, která je udávána v bitech. Například 24 bitová RGB barva, která obsahuje 8 bitů na každou barevnou složku, jedna barva je reprezentována 8 bity, což je 256 odstínů barvy. U zobrazení RGB modelu se můžeme setkat s vyšším zastoupením zelené barvy a nižším zastoupením modré barvy. Je to z důvodu citlivosti lidského oka, které je nejvíce citlivé na zelenou barvu a nejméně na barvu modrou.

### 3.2.1 Stupně šedi

Pokud se budu chtít zbavit barev na obraze, je dobré využít filtr, který umožňuje převedení obrázku na stupně šedi [3]. Nejčastěji se převádí obraz na 256 odstínů šedé, tudíž na 8 bitovou hloubku. Převádění je prováděno podle barevného zastoupení, nebo podle vlastní volby.

Na konci stránky je 6 obrázků (Obr. 10), na kterých je srovnáváno převádění na stupně šedi. První obrázek je originální, na druhém je znázorněno převedení podle zastoupení červené barvy v obrázku, na třetím podle zelené barvy a na čtvrtém podle modré. U pátého obrázku jsou barvy pro každý pixel zprůměrovány vzorcem $(R+G+B)/3$. Na poslední obrázek je použit doporučený poměr při převádění na stupně šedi se zastoupením 30 % červené, 59 % zelené a 11 % modré. Tento doporučený poměr také použiji ve své práci, protože se zdá být nejvhodnější.

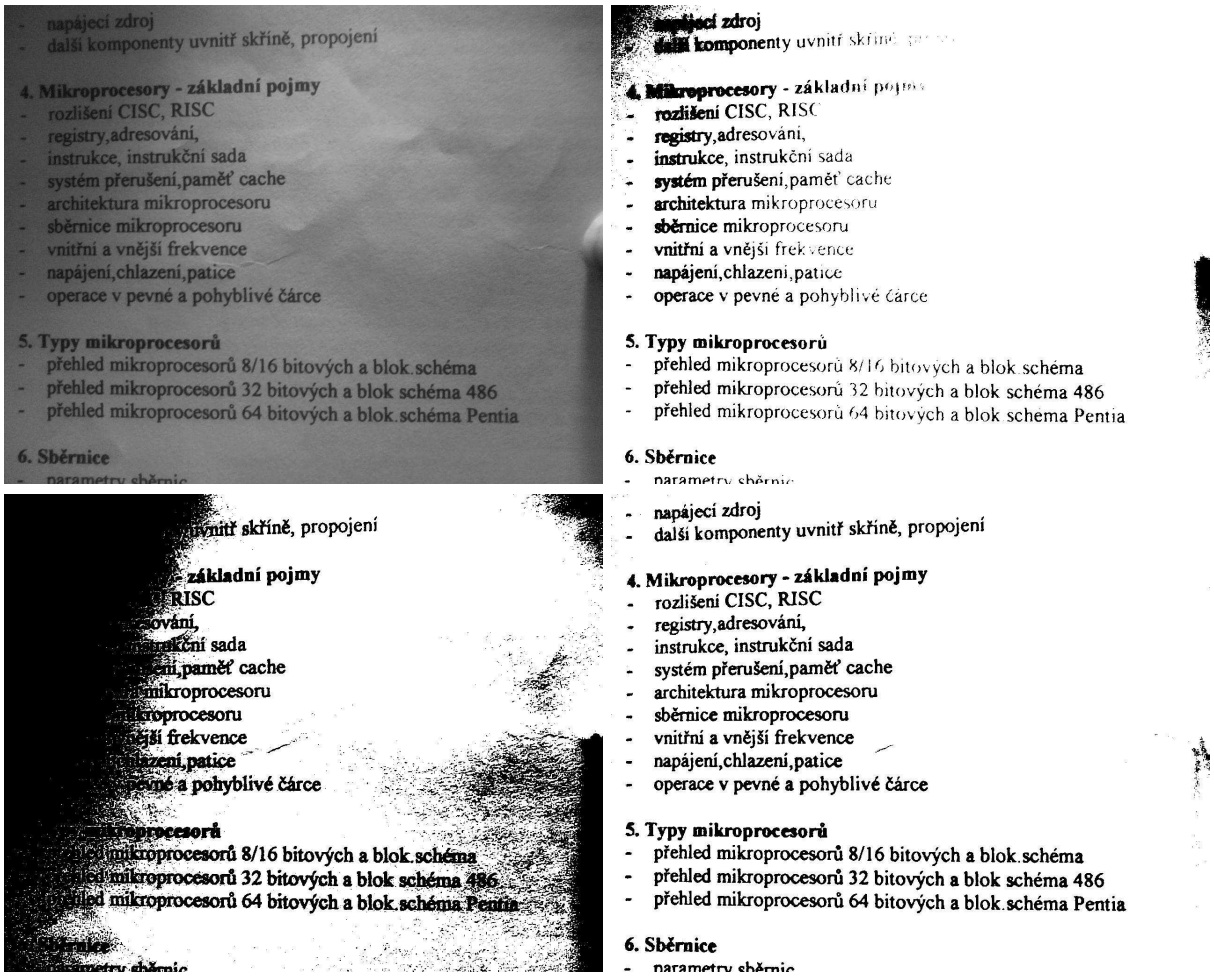
Obrázek 10: Originální obrázek, převod na šedou přes R, G, B, zprůměrování všech složek, doporučené hodnoty: R = 30, G = 59, B = 11

### 3.2.2 Prahování

Po úspěšném převedení obrázku na stupně šedi, bych potřeboval rozlišit černý text a bílé pozadí. K tomuto účelu se nejlépe hodí prahování, neboli převod na binární obraz [2]. Prahování zajišťuje, že každý pixel výsledného obrázku bude pouze černý, nebo bílý. Žádná jiná barva nebo odstín mezi tím, pokud ovšem nechceme použít jiné dvě barvy. Prahování funguje tak, že si vezmeme pixel s polohou $x, y$ v obrázku, nejlépe v stupních šedi a vybereme jeho intezitu jasu. Pokud je intenzita vyšší než stanovená hranice prahování, pixel získá barvu bílou, pokud je nižší, pixel bude černý. Konkrétněji, pokud máme obrázek ve stupních šedi s 8 bitovou hloubkou, nějaký bod A se souřadnicemi $x, y$ a jeho intenzitu jasu označenou $I$, pak už jen schází nastavit hodnotu prahování $\tau$ od 0 do 255 a vše dosadit do vzorce. Pokud vše provedeme pro všechny body obrázku, vznikne nám nový obrázek pouze s černou a bílou barvou (Obr. 11).

$$A_I(x,y) = \begin{cases} 1 & \text{if } I(x,y) \geq \tau \\ 0 & \text{if } I(x,y) < \tau \end{cases} \tag{9}$$

Prahování bodu $A_I(x,y)$, kde $A_I$ je intenzita jasu bodu $A$ a $\tau$ je hranice prahování




Obrázek 11: Originální obrázek, globální prahování - ruční nastavení prahu, globální prahování - automatické nastavení (průměrování), lokální prahování Bradley

Prahování se dá rozdělit na tři způsoby provedení: prvním způsobem je prahování globální, kdy se na celý dokument použije stejná hranice prahování $\tau$. Zástupci tohoto způsobu prahování jsou metody, které počítají hranici prahování ze všech bodů, například díky nalezení maximální a minimální intenzity jasu, použitím histogramu pro nalezení hranice, či prostého zprůměrování. Druhým způsobem je takzvané poloprahování, kdy se prahování použije pouze na určité intenzity jasu bodu, avšak výsledný obraz nám zůstává ve stupních šedi. Posledním způsobem je adaptivní, neboli lokální prahování. Tato metoda se použije, pokud se v obraze vyskytují různé stíny, odlesky či jiné anomálie a není vhodné použít globální prahování. Existuje několik způsobů řešení, například metod SIS, Bradley nebo metoda Otsu.

### 3.2.3 Jas a kontrast

Jednou z nejběžnějších úprav obrazu, se kterou se můžeme setkat, je úprava jasu [11]. U klasického barevného obrázku ve formátu RGB se jas skládá z intenzity všech tří barev. Nejčastěji se s úpravou jasu setkáme, když máme obraz příliš tmavý, nebo naopak příliš světlý. Vezmu si například osmibitový obrázek ve formátu RGB, který má na každou složku 256 odstínů barvy, tak barva bodu reprezentována samými nulami, bude v tomto formátu černá a naopak barva tvořena u každé složky hodnotou 255, bude bílá.

Klasická verze úpravy jasu funguje tak, že obrázek je zesvětlován, tudíž všechny body obrazu jsou zesvětlovány stejnou hodnotou, neboli je přidávána ke všem kanálům barev stejná hodnota. Naopak pokud je obrázek ztmavován, je stejná hodnota u všech kanálů odebírána. Když už není možnost u daného pixelu odebrat či přidat intenzitu kanálu, tato složka zůstává beze změny. Mimo klasické nastavení jasu se můžeme setkat s pokročilejší metodou, například pomocí Gaussovy funkce. Existují také barevné modely v počítačové grafice, které mají vlastní složku, tvořenou pouze jasem. Největší výhodou těchto formátů je úprava jasové a barevné složky zvlášť. Jsou to například modely ARGB, HSL a další.

Druhou zmíněnou úpravou je nastavení, neboli úprava kontrastu [11]. Úprava kontrastu funguje tak, že se zvyšuje či snižuje rozdíl mezi tmavými a světlými částmi obrazu. U globálního kontrastu je proveden výpočet střední šedé barvy a všechny pixely, které mají intenzitu nižší, to znamená, že jsou tmavší, posuneme směrem k černé. Naopak světlejší se posunou směrem k bílé barvě. Dalším ze způsobů je použití takzvaného dynamického kontrastu, použitím Gaussovy funkce nebo lokálního kontrastu či kontrastu pouze určité barvy. V profesionálních programech se na úpravu kontrastu nebo také jasu nejčastěji využívají histogramy. Jednou z dalších možností, jak upravit světlost obrazu je použití gamma korekce, která mění pouze středně tmavé body, další variantou je použití logaritmické transformace.

Obrázek 12: Originální obrázek, mírná úprava jasu (zesvětlení), úprava kontrastu

### 3.2.4 Potlačení šumu

Nejen při nedostatku světla se na zachycené fotografii s velkou pravděpodobností objeví šum [6]. Šum je vlastně anomálie, která může mít podobu zrnění na obraze a vzniká náhodně a to nejčastěji díky nedostatku světla dopadajícího na fotocitlivý čip nebo při nastavení příliš velké citlivosti čipu. Šumu se úplně nelze zbavit, ale lze jej eliminovat dobře nasvíceným objektem či scénou, více šumu lze logicky očekávat na tmavých místech fotografie. Většina zařízení umožňuje potlačit šum již před zachycením fotografie. Z rastrového formátu lze šum částečně eliminovat pomocí filtrů. Filtry na potlačení šumu můžeme rozdělit na dva typy, lineární a nelineární. Lineární rozmazávají obraz, ale my potřebujeme text zaostřit, proto jsou vhodnější v našem případě filtry nelineární.

Jeden z nejznámějších nelineárních filtrů je median (Obr. 13). Tento filtr funguje tak, že vybere hodnoty intenzity jasu okolních bodů zvoleného pixelu, tyto hodnoty seřadí a vybere z nich prostřední hodnotu. Vybranou hodnotu dosadí na místo vybraného pixelu. Může se zdát, že tato metoda musí vyrušit obraz, ale většina bodů na obraze jsou si podobných a median je velice účinná metoda na potlačení extrémů, neboli šumu či jiných drobných anomálií, které se mohou vyskytovat v obraze.

Další metodou, kterou lze k potlačení šumu použít, je metoda konzervativního vyhlazení, která obdobně jako median vybírá z okolních bodů vybraného pixelu, avšak tentokrát se vybírá maximální a minimální hodnota intenzity jasu okolních bodů. Po nalezení těchto dvou hodnot se hodnoty porovnají s intenzitou jasu vybraného pixelu. Pokud je intenzita vybraného pixelu vyšší než maximální hodnota z okolních bodů (11), dosadí se místo vybraného bodu bod s maximální hodnotou. To samé se provede, pokud je hodnota vybraného bodu nižší než minimální hodnota (10), avšak se dosadí bod s minimální hodnotou. Pokud hodnota jasu je mezi minimem a maximem, žádné výměny se neprovádí a pokračuje se bodem následujícím.

$$A_I(x,y) < minI \Rightarrow A_I(x,y) = minI \tag{10}$$

$$A_I(x,y) > maxI \Rightarrow A_I(x,y) = maxI \tag{11}$$

Konzervativní vyhlazení, $A_I$ - intenzita jasu bodu $A$, $(x,y)$ - souřadnice bodu , $minI$ - minimální intenzita okolního bodu, $maxI$ - maximální intenzita okolního bodu

Obrázek 13: Původní obrázek, obrázek po použití medianu

## 3.3 OCR

Optické rozpoznávání znaků (Optical Character Recognition - OCR) [12] je služba, která se snaží převést kopii textového souboru ve formě rastrového obrázku do podoby textu ve znacích. Hlavní výhodou OCR je rychlost, kdy přeložený dokument pomocí OCR je vygenerovaný během několika sekund, zatímco ruční přepisování textu by zabralo několik minut či desítek minut. Nevýhodou optického rozpoznávání znaků je nutná dodatečná ruční kontrola a korekce. Rastrový obrázek může být naskenovaný pomocí skeneru, nebo třeba vyfotografovaný fotoaparátem, webkamerou či jiným zařízením, které je schopné zachytit obraz. Program, který OCR používá, se nejdříve pokusí analyzovat obrázek s textem a vybrat jednotlivé řádky a písmena textu. Protože jsou řádky i písmena od sebe odděleny prostorem, neměla by být detekce jednotlivých znaků černého textu na bílém podkladu problémem.

Největším problémem je samotné rozpoznání jednotlivých znaků, protože v současnosti existuje nespočet typů písma, zatímco v historii existovaly jen dva typy písma, které bylo možno po zkopírování do elektronické podoby přeložit zpět do textové podoby. Tím byly standardy OCR-A a OCR-B. Standard OCR-A byl vyvinut jako první a používán v USA. Standard OCR-B byl vyvinut pro Evropu jako reakce na typ A užívaný v zámoří.

Tento text je napsán ve standardu OCR-B

Ukázka textu ve standardu OCR-B

V současné době již standardy nefungují a spoléhá se na nejpoužívanější a nejlépe čitelné typy písma. Avšak problémem je stále identifikace jednotlivých znaků v textu, navíc existuje mnoho písmen či jiných znaků, které jsou si velice podobné, například často užívané velké písmeno O a číslo 0. Proto se u některých typů písma v textu můžeme setkat s nulou, která je uvnitř napříč přeškrtlá, nebo obsahuje uprostřed vnitřního prostoru tečku.

Jedním z aspektů každého znaku jsou jeho jedinečné tvary, které se využívají u jednoho z typů identifikace znaků, kde každý znak zabírá trochu jiné body v prostoru, na kterém je zobrazen. Poté už stačí jen využít algoritmus, který bude prostor, jenž znak zabírá, analyzovat a vyhodnocovat. Druhým typem identifikace je použití databáze předloh jednotlivých písmen a využití srovnání a následné procentuální shody identifikovaného znaku se znaky z databáze. Tato metoda se dle mého názoru zdá být účinnější z důvodu existence mnoha typů písma.

Dnes existuje na trhu mnoho více či méně kvalitních komerčních i nekomerčních produktů, které s metodou rozpoznávání znaků umí velice dobře pracovat. Téměř ke každému skeneru výrobci přidávají software, který umí naskenovaný text přímo převést do textového editoru. Také existuje mnoho desktopových aplikací pracujících s OCR a v neposlední řadě je tu několik webových stránek, umožňujících konverzi textu z obrázku na čistý text i s podporou českých znaků a to bez nutnosti instalace nějakého softwaru. Avšak stále je pro tyto produkty nezbytné mít kvalitní vstupní obrázkový dokument s co možná nejvyšším rozlišením a nejlépe čitelným textem. I přes kvalitně zachycený dokument je velice pravděpodobné, že se ve výsledném textu objeví chyby, které je nutné ručně opravit.

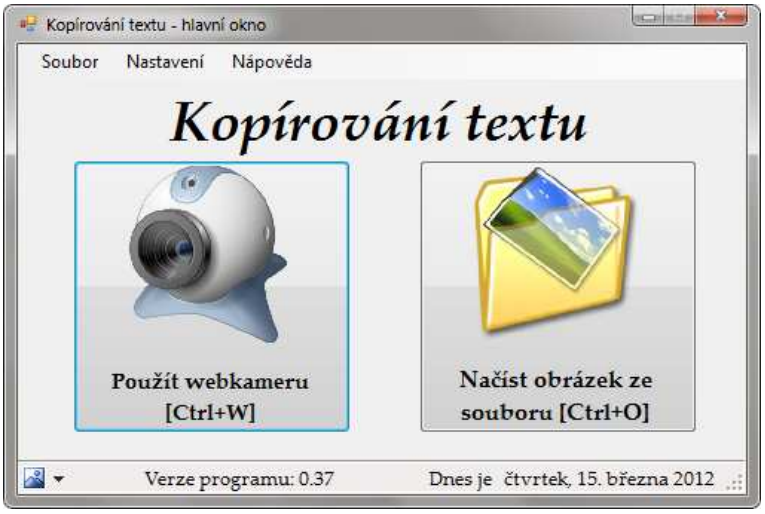
# 4 Řešení

Ve své bakalářské práci mám za úkol vytvořit program, který bude pomocí webkamery schopen vytvořit kopii dokumentu. Nastává zde několik problémů, které jsem se pokusil vyřešit. Ke své programové části jsem si zvolil psaní kódu v programovacím jazyku C#, jedná se o objektově orientovaný jazyk, který běží na platfomě .NET. K tomuto jazyku jsem si vybral nástavbu pro zpracování obrazu - jedná se o knihovny pro zpracování obrazu, matematiky a robotiky s názvem AForge.NET [13]. Z těchto balíčků budu používat pouze některé knihovny, které budou nezbytné pro mou práci. Konkrétně pro video a obsluhu webkamery budu potřebovat knihovny AForge.Video a AForge.Video.DirectShow a pro práci se zobrazením, obrazem a obrazovými filtry knihovny AForge.Imaging a AForge.Imaging.Filters. Pomocí těchto knihoven budu moci využívat již napsané třídy pro zpracování obrazu a videa, které obsahují. Dle mého názoru je výhodné používat knihovny a funkce vestavěné a přidané k danému programovacímu jazyku, protože jsou optimalizované pro běh v daném jazyku a v naprosté většině jsou rychlejší než pozdější implementace v kódu.

## 4.1 Úvodní okno programu

Po spuštění programu jsem vytvořil hlavní třídu a pojmenoval ji hlavní okno (Obr. 14), která je považována za rozcestník mezi použitím webkamery a úpravou fotografie. Tento rozcestník je tvořen oknem s jednoduchým menu, ve kterém jsou umístěny položky pro práci s formulářem, položky nastavení a nápovědy. K většině položek je možné také přistupovat pomocí klávesových zkratek. Hlavním důvodem vytvoření hlavního okna byla možnost výběru práce s programem.

Program umožňuje v tomto formuláři vyvolat okno, které pracuje s webkamerou a tudíž používá knihovny pro práci s videem. Druhou variantou je možnost vyvolat formulář výběru obrázku ze souboru na disku, s následnou úpravou zvoleného obrázku. Důvodem takovéto koncepce programu je jednoduchý výběr obrázku z disku, není nutné proklikávat se přes celý program, když například není k dispozici webkamera. Pohodlně zvolíme z hlavní obrazovky otevření obrázkového souboru, popřípadě použití webkamery. Hlavní okno programu obsahuje jednoduché hlavní menu, dvě tlačítka (použití webkamery a otevření obrázku) a statusbar.



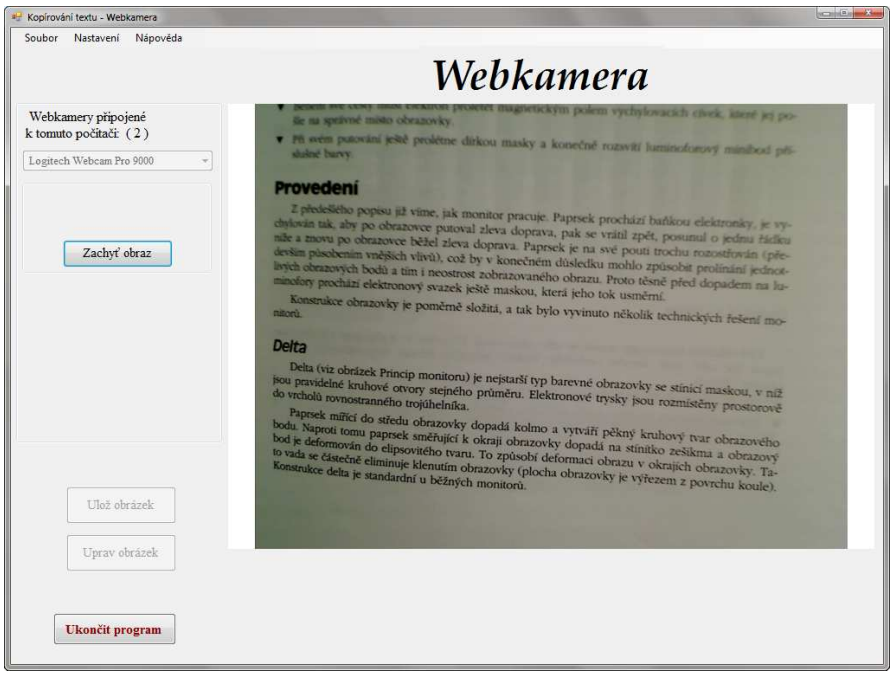

Obrázek 14: Hlavní okno programu vyvolané po spuštění programu

## 4.2 Získání obrazu z webkamery

Vytvořil jsem třídu, která řeší zachytávání obrazu a umožnuje obsluhovat webkamery připojené k počítači. Při vyvolání okna obsluhy program automaticky prohledá dostupné přístroje a poté je nabídne k výběru. Pokud jsou ovladače webkamery správně nainstalovány, neměl by být problém s její identifikací v programu. Aplikace umožňuje automatické prohledání podporovaných rozlišení vybrané webkamery a pokud to zařízení umožňuje, tak i rozlišení fotoaparátu. Když zařízení nepodporuje funkci fotoaparátu, je v aplikaci řešeno zachycení obrázku z videa.

Po kliknutí na tlačítko "Zachyť obraz"se na plátně zobrazí zachycený obrázek. Ač se tak může zdát, tak to není úplně pravda, protože existují dvě plátna, přičemž na jedno se promítá aktuální snímané video, na druhé je zobrazen zachycený obrázek a plátna jsou jen střídavě zviditelňována. Je to jednodušší a efektivnější řešení než stálé vypínání a zapínání webkamery. Webkamera je automaticky vypnuta, pokud dojde k uzavření programu, nebo otevření jiného okna. Základní verze nastavení umožňuje jednoduchou manipulaci s výběrem webkamery, nastavení jednotlivých rozlišení a dalších funkcí je skryto a umožněno po zvolení pokročilých nastavení. Automaticky je u webkamery nastaveno nejvyšší rozlišení, protože pro základní používaní není nutná změna této volby.

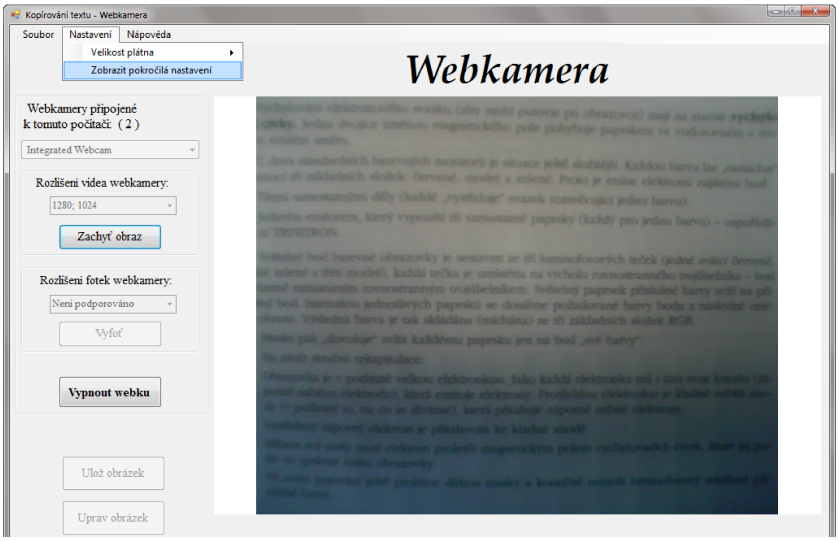
Nepostradatelnou funkcí je vyhledání připojených zařízení až po spuštění okna webkamery a to z důvodu jeho pozdějšího připojení k počítači. Tuto funkci jsem vyřešil tlačítkem "Aktualizuj", které je viditelné v základním režimu pouze před zapnutím webkamery.




Obrázek 15: Základní okno pro obsluhu připojených webkamer

Formulář webkamery (Obr. 15) obsahuje tlačítko pro zapnutí webkamery, tudíž také start promítání snímaného videa na plátno. Tlačítko aktualizace připojených zařízení. Po zachycení obrazu jsou zpřístupněna tlačítka pro uložení a editaci obrázku, přičemž tyto funkce lze také vyvolat přes menu, nebo klávesovou zkratku.

Pro nepřehlednost původního formuláře jsem vytvořil dvě volby práce s webkamerami, kde základní, jednodušší varianta je načtena automaticky a pokročilá volba nastavení se zobrazí až po zvolení pokročilých nastavení z hlavního menu (Obr. 16). Pokročilá volba umožňuje nastavení jednotlivých rozlišení webkamery a umožňuje zvlášť vyfotit či zachytit obraz z videa. Další možností, kterou pokročilá volba umožňuje, je zapínání a vypínání webkamery, což nám dává možnost pozdějšího výběru a aktualizaci webkamer.




Obrázek 16: Pokročilé nastavení webkamer

Kousek zdrojového kódu metody "VyhledejWebky"třídy webkamera, na němž je řešeno vyhledávání webkamer a následné naplnění seznamu připojenými zařízeními.

```
                                                    ...
public void VyhledejWebky()  // metoda která prohledá dostupné zařízení
{
   try
   {
      WebZarizeni = new FilterInfoCollection(FilterCategory.VideoInputDevice);
      cbPripojenaZarizeni.Items.Clear();
      if (WebZarizeni.Count == 0)  // Nebyla žádná webka nalezena
      {
         WebkaNalezena = false;  // boolean se nastaví na false
         switch (MessageBox.Show("Nenalezeno žádné zařízení, zkuste vyhledat zařízení znovu",
              "Webkamera nebyla nalezena", MessageBoxButtons.RetryCancel, MessageBoxIcon.Hand))
                                                    ...
      }
      else  // Nalezeno
      {
         WebkaNalezena = true;  // boolean se nastaví na true
         BuZapni.Enabled = true;  // zobrazí se tlačítko pro zapnutí webkamery
         cbRozliseni.Enabled = true; cbFotoRozliseni.Enabled = true;
         foreach (FilterInfo device in WebZarizeni)
         {
            cbPripojenaZarizeni.Items.Add(" " + device.Name);
         }
         video = new VideoCaptureDevice(WebZarizeni[0].MonikerString);
         cbPripojenaZarizeni.SelectedIndex = 0;  // Defaultní výběr zařízení
         laVyberWebku.Text ="Webkamery připojené\n k tomuto počítači: "+" ( "+WebZarizeni.Count+" ) ";
         RozliseniZarizeni();
      }
   }
                                                    ...
```

Následující kód znázorňuje naplnění seznamu rozlišení videa webkamery, poté vyhledání a nastavení nejvyššího podporovaného rozlišení. To samé jako s videem se provede i pro fotoaparát, pokud jej webkamera podporuje.

```
                                                ...
videoRozliseni = video.VideoCapabilities;  // rozlišení videa
fotoRozliseni = video.SnapshotCapabilities;  // rozlišení fotoaparátu
cbRozliseni.Items.Clear();  // musíme vyčistit jinak se budou rozlišení přidávat
cbFotoRozliseni.Items.Clear();
int maximalni = 0;
foreach (VideoCapabilities polozka in videoRozliseni)
{
   cbRozliseni.Items.Add(polozka.FrameSize);  // přidání rozlišení
   if (polozka.FrameSize.Width >= maximalni)   // nastaví nejvyšší rozlišení
   {
      maximalni = polozka.FrameSize.Width;
      cbRozliseni.SelectedIndex = cbRozliseni.Items.Count - 1;
   }
}
                                                ...
```

U další části kódu, která pochází z metody zapínání videa, se využívá vybraných položek ze seznamů webkamer a jejich rozlišení. Zjišťuje se zda webkamera umožňuje fotografovat a nastaví se pro ni také rozlišení. Pro vytvoření videa na plátně se využívá funkce klonování obrazu videa.

```
                                                ...
ZdrojVidea = new VideoCaptureDevice(WebZarizeni[cbPripojenaZarizeni.SelectedIndex].MonikerString); //
    vybere zvolenou webku -MonikerString!! pro identifikaci
pbObraz.Image = null;   // vymazani pictureboxu
ZdrojVidea.NewFrame += new NewFrameEventHandler(NovyObrazVidea);  // klonování obrazu
UkoncitSnimani();
ZdrojVidea.DesiredFrameSize = videoRozliseni[cbRozliseni.SelectedIndex].FrameSize;  // vybere zvolené
    rozlišení
buZachyt.Enabled = true;
if ((cbFotoRozliseni.Items.Count > 0) && (cbRozliseni.Enabled == true))  // zda umožňuje fotit
{
   buVyfot.Enabled = true;
   ZdrojVidea.ProvideSnapshots = true;  // umožněno zachycení snímků
   ZdrojVidea.DesiredSnapshotSize = fotoRozliseni[cbFotoRozliseni.SelectedIndex].FrameSize;
   ZdrojVidea.SnapshotFrame += new NewFrameEventHandler(NovyObrazFoto);
}
ZdrojVidea.Start();  // zapnutí streamu videa
                                                ...
```

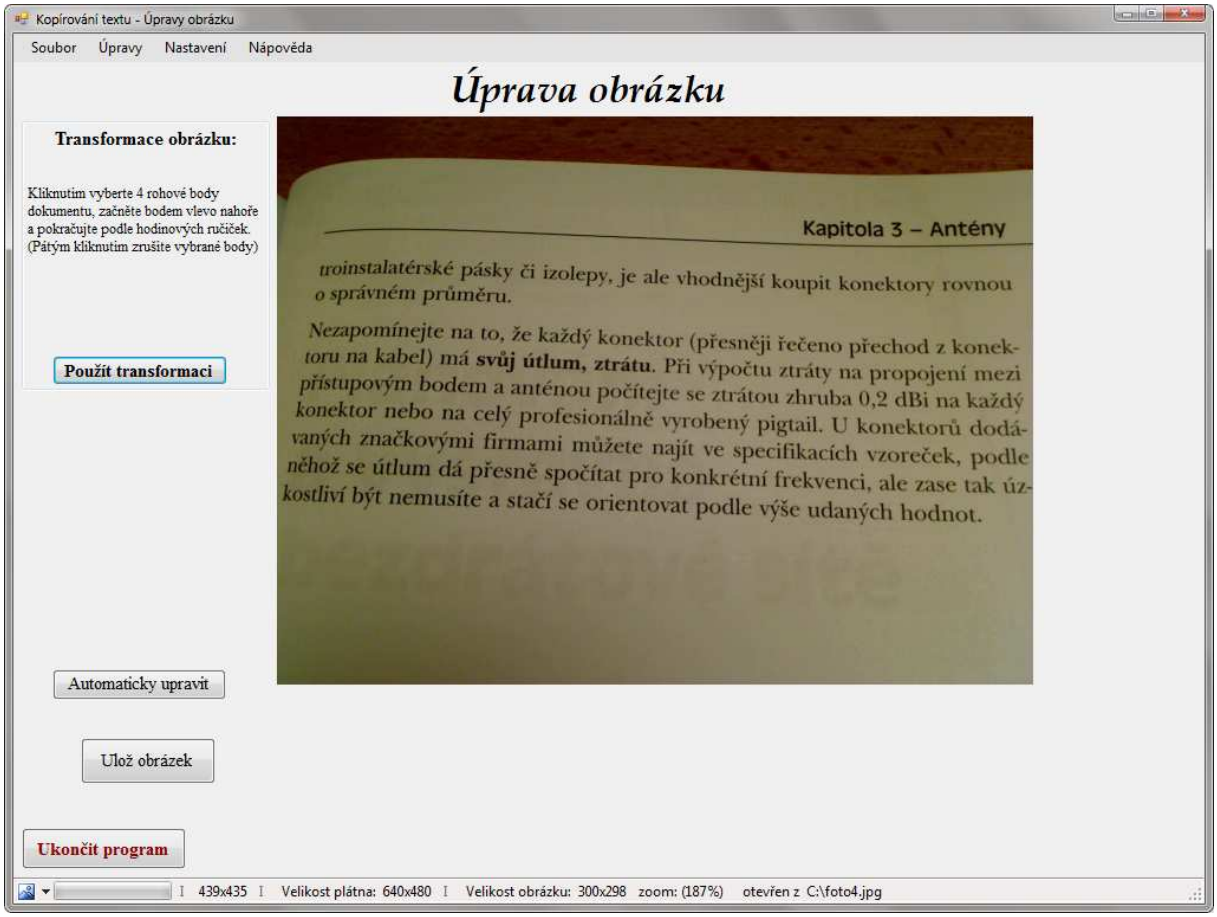
Velmi důležité před zavřením programu nebo před přepnutím okna do úprav, je ukončení chodu webkamery, protože by se mohlo stát, že po zavření programu by webkamera stále běžela v procesech a nebylo by možné z ní následně získat obraz. Proto jsem vytvořil metodu, která po zavolání zjistí, zda webkamera běží a případně ji ukončí.

```
                                                ...
private void UkoncitSnimani()  // metoda která ukončí stream webky
{
   if (ZdrojVidea != null) // pokud video běží
   {
      ZdrojVidea.SignalToStop();  // Zastav
      ZdrojVidea.Stop();
   }
}
                                                ...
```

## 4.3 Úprava obrazu

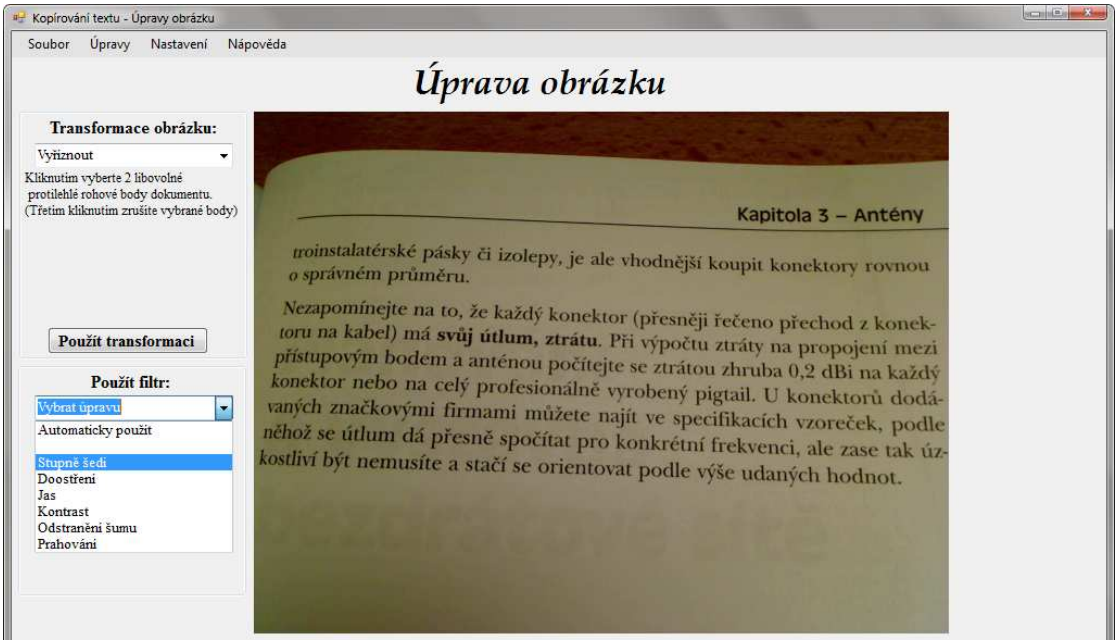
Dokument, který se mi podařilo zachytit, bych měl ještě upravit, aby byl co nejlépe čitelný. K úpravě vyfotografovaného dokumentu jsem využil základních geometrických transformací a některých filtrů vhodných pro úpravu čitelnosti. Směrodatným faktorem se stává oddělení samotného textu od okolních elementů, které nepatří k textu a následné jasné rozdělení dokumentu na text a na jeho pozadí.

Nové okno pro úpravu fotografie je nastaveno v základním, přehlednějším režimu (Obr. 17). Stejně jako tomu bylo u okna webkamery, lze také zapnout pokročilejší nastavení. V základním nastavení program umožňuje oříznutí a narovnání obrázku z prostoru, přičemž aplikace využívá čtyřúhelníkovou transformaci, u které je nutné znát čtyři rohové body dokumentu. Tyto body lze jednoduše vybrat klikáním myši na oblast obrázku. Po této úpravě transformace, lze zvolením tlačítka "Automaticky upravit", využít přednastavené filtry, které se snaží odfiltrovat nežádoucí efekty na obraze a jednoznačně oddělit text od papíru. U některých obrázků je lepší použít volby obráceně, nejdříve použít filtry a následně obraz vyříznout. Kvůli možnosti nechtěné úpravy lze použít historii a vrátit stav do původního zobrazení. Statusbar v zápatí stránky zobrazuje důležité zakladní informace o obrázku jako je poloha myši na plátně, velikost obrázku, či zoom obrázku na plátně.




Obrázek 17: Úprava obrázku - základní nastavení

Vybráním volby pokročilé nastavení (Obr. 18) se zobrazí seznamy jednotlivých transformací a filtrů. Tyto úpravy je nyní možné vybírat jednotlivě a nastavovat podle vlastního uvážení, přičemž téměř každá změna, která je provedena se téměř ihned zobrazí na plátně a pro aplikování je nutné potvrdit tlačítkem "použit transformaci"nebo "použít filtr". V opačném případě nebudou změny uloženy a program bude pokračovat v úpravách s původním obrázkem. Při nechtěném potvrzení změny nebo změně rozhodnutí, je k dispozici možnost vzít krok zpět, přičemž nejen jeden, ale maximálně devět kroků do historie. Historie změn je jednoduché pole, do kterého se ukládají obrázky, tyto obrázky lze následně přes menu, nebo klávesovou zkratkou v krocích vyvolávat. Dle potřeby se u úprav, ať už pro transformace nebo filtry, zobrazí posuvníky či zaškrtávací položky, pomocí nichž se budou provádět potřebné úpravy.




Obrázek 18: Úprava obrázku - pokročilé nastavení

Pro celý formulář úprav jsem vytvořil dvě proměnné (bitmapy), které v sobě uchovávají buďto promítaný obrázek nebo obrázek, na kterém jsou prováděny úpravy. Pro předání vyfotografovaného obrázku pomocí webkamery, nebo otevření obrázku z hlavního menu se z těchto tříd zavolá metoda "ZobrazObrZeZdroje", která zobrazí obrázek na plátně, uloží jej do historie a nastaví okno úprav do základního režimu práce s obrazem.

```
                                                ...
public partial class Upravy : Form
{
   private Bitmap bit = null;  // proměnná do které se ukládá potvrzený obrázek
   private Bitmap bit2 = null;  // proměnná ve které je aktuálně promítaný obrázek
                                                ...
   public void ZobrazObrZeZdroje(Bitmap pic)  // Obrázek zachycený z webky, nebo otevřený ze souboru
   {
      pbPlatno.Image = pic;  // Na plátno se promítne obrázek
      bit = pic;  // Do proměného bitmapu se uloží obrázek
      pbPlatno.SizeMode = PictureBoxSizeMode.Zoom;
      ZobrazPrvkyDefault();  // Zobrazí se defaultní nastavení
      UlozDoHistorie(bit);
   }
                                                ...
```

Na následujícím kusu kódu je znázorněna práce s metodou transformace z prostoru, nazvanou "z3Ddo2D". Protože otevřený obrázek nemusí mít stejný poměr stran jako plátno, na který je promítán, je nutné dopočítat bod obrázku z bodu kliknutí na plátně. K tomuto účelu jsem vytvořil metodu "PomerObrazku", která vrátí ze vstupního obrázku a velikosti plátna poměr výšky k výšce plátna a šířky k šířce plátna. Poté jednoduše odečteme od souřadnic vybraného bodu prázdný prostor před obrázkem na plátně a vynásobíme poměrem, tímto způsobem získáme všechny požadované body. A nyní tyto body stačí vhodně použít do transformace.

```
                                                ...
private double[] PomerObrazku(Bitmap obr) ...

private Bitmap Vyrez(Bitmap obr) ...

private Bitmap z3Ddo2D(Bitmap obr)
{
   try
   {
      Bitmap obr2 = obr;
      double[] pomery = PomerObrazku(obr2);
      List<IntPoint> Body4 = new List<IntPoint>();

      int aX = Convert.ToInt32((rohoveBody4[0] - pomery[2]) * pomery[0]);  // odečítáme prostor na
          začátku, násobíme poměrem šířka obr/ šířka plátna
      int aY = Convert.ToInt32((rohoveBody4[1] - pomery[3]) * pomery[1]);
      int bX = Convert.ToInt32((rohoveBody4[2] - pomery[2]) * pomery[0]);
                                                ...

      QuadrilateralTransformation transformace = new QuadrilateralTransformation(Body4, bod1x, bod1y);
      obr2 = transformace.Apply(obr);
      return obr2;
}
                                                ...
```

Po stisknutí tlačítka automatické úpravy se zavolá metoda "AutomatickyUprav", která má na starosti z původního obrázku vytáhnout text dle přednastavených filtrů. U následujícího kódu je znázorněna část této metody, jsou zde volány metody na změnu indikace průběhu zpracování obrázku a také samotné metody úprav. Vše je ukončeno zobrazením výsledného obrázku na plátně a uložením do historie.

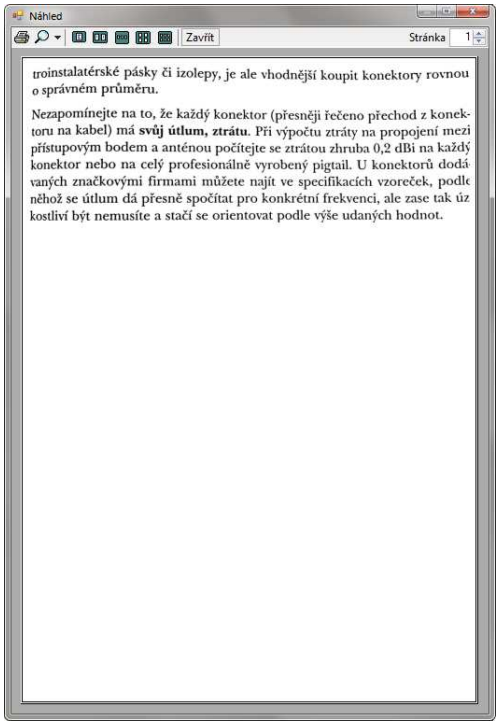
```
                                                ...
private void AutomatickyUprav()
{
   ZobrazPrvky();  // prvky se nastaví do defaulní polohy
   KurzorZmena();  // nastaví se kurzor na wait
   ProgressZmena(15);  // progressbar značí postup zpracování

   tbKernel.Value = 6;
   tbSigma.Value = 35;
   bit2 = ZostreniGausovske(bit);  // při automatickém paužití se pužijí přednastavená kombinace metod
   ProgressZmena(30);

   tbKernel.Value = 2; bit2 = Kontrast(bit2);
   ProgressZmena(45);
                                                ...

   pbPlatno.Image = bit2;  // zobrazí se na plátně
   bit = bit2;
   UlozDoHistorie(bit2);  // ukládá se obrázek do historie
   ProgressZmena(0);
   KurzorZmena();
}
                                                ...
```

Upravený obrázek je možné v programu uložit, nebo jej přímo vytisknout. Vybráním volby menu "Náhled tisku"(Obr. 19), program vyvolá nové okno s aktuálním náhledem upraveného obrázku na papíře. Obrázek lze také přímo vytisknout bez zobrazování náhledu a to přes volbu hlavního menu "Tisknout", nebo pomocí klávesové zkratky.




Obrázek 19: Okno náhledu tisku upraveného obrázku

Tato část kódu programu řeší zobrazení náhledu obrázku, je zde nastavena velikost okna náhledu a také přednastaven název obrázku. Volba náhledu tisku využívá metodu "ZdrojTisku", ve které je měněna velikost obrázku tak, aby se obrázek přizpůsobil papíru a nebyl oříznut.

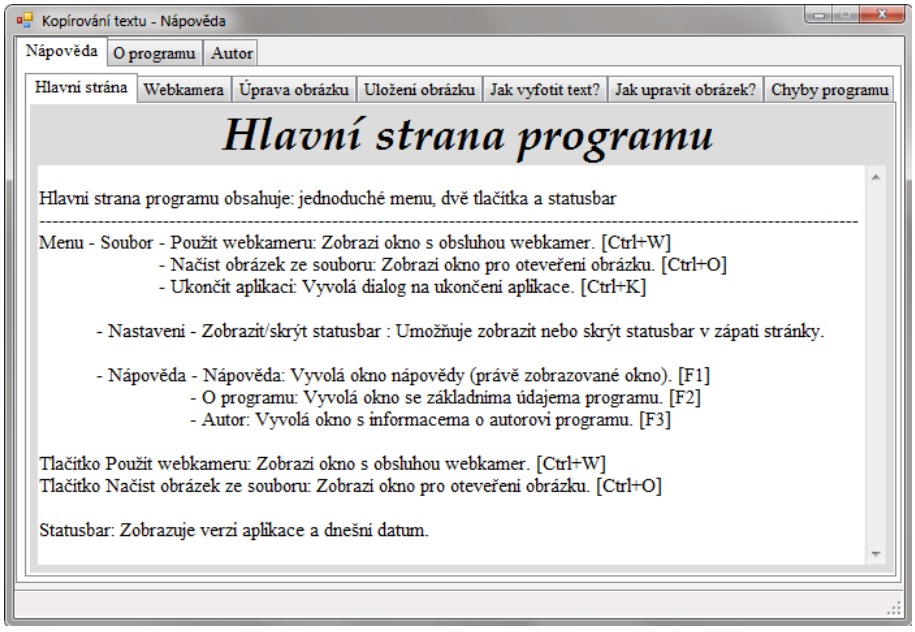
```
                                                    ...
private void miNahledTisku_Click(object sender, EventArgs e)
{
   try
   {
      pdNaVytisknuti.PrintPage += new PrintPageEventHandler(ZdrojTisku);
      ppdNahledTisku = new PrintPreviewDialog();
      ppdNahledTisku.Document = pdNaVytisknuti;
      ppdNahledTisku.Height = 600; // velikost okna tisku
      ppdNahledTisku.Width = 400;
      ppdNahledTisku.Document.DocumentName = "Obrázek"; // název tisknutého obrázku
      ppdNahledTisku.Show();
   }
   catch (Exception)
   {
      MessageBox.Show("Nepodařil se otevřít náhled tisku obrázku", "Chyba náhledu tisku",
          MessageBoxButtons.OK, MessageBoxIcon.Warning);
   }
}

private void miTisknout_Click(object sender, EventArgs e) ...
                                                    ...
```

## 4.4 Nápověda k programu

Součástí celého programu je jednoduchá nápověda (Obr. 20) tvořená několika záložkami, které popisují práci s programem. Současně s nápovědou existují záložky: O programu (specifikace požadavků a funkcí) a Autor (informace o autorovi programu). Nápověda je vytvořena v textovém souboru (Nápověda.help) a je vygenerována společně s knihovnami u aplikace. Program je napsán tak, že při chybějícím souboru nápovědy program v pořádku funguje, avšak při vyvolání nápovědy je zobrazena varovná hláška a záložky jsou bez textu. Ukázka části kódu, na kterém je znázorněno řešení čtení nápovědy z textového souboru:

```
                                                    ...
FileStream fs = new FileStream("Nápověda.help", FileMode.Open);  // Otevření souboru nápovědy
StreamReader sr = new StreamReader(fs);
string radek; int index = 0; Boolean zapis = true;
while ((radek = sr.ReadLine()) != null)
{
    if (radek.Length > 2)
   {
       if (radek.Substring(0, 3) == "###")  // označení oddělení částí nápovědy
      {
         index++;
         zapis = false;
      }
      else
      {
         zapis = true;
      }
   }
   if (zapis == true)  // pokud slovo nezačínu 3 křižkama zapiš
                    {
                        switch (index)  // Rozdělení zápisu podle indexu
                        {
                            case 1:  // Index 1 zapiš do první záložky
                                rtbHlavniStrana.Text = rtbHlavniStrana.Text + "\n" + radek;
                                break;
                                                    ...
```




Obrázek 20: Okno nápovědy - hlavní strana

## 4.5 Výstupní obrázky

Uložit obrázek lze přes tlačítko, položku hlavního menu, či klávesovou zkratku. Systém vyvolá nové okno uložení obrázku (Obr. 21) s výběrem formátu a aktuálním náhledem ukládaného obrázku. Pro ukládání jsem vytvořil novou třídu s názvem "Ulozit.cs", protože je uložení možné vyvolat jak z okna webkamery, tak z okna úprav.

Aktuální verze programu umožňuje uložit obrázek ve formátu bmp, jpg a png. Přičemž primární formát je formát png, tento formát obrázku se stává kompromisem mezi formátem bmp, jenž je náročný na velikost a formátem jpg, u kterého se projevuje neblahý vliv ztrátové komprese (šum kolem okrajů objektů v obraze). V zápatí okna ukládání je statusbar, který informuje o stavu uložení obrázku. Formulář obsahuje dvě tlačítka, přičemž přes tlačítko "Ulož obrázek"program vyvolá dialog, přes který si uživatel zvolí cestu uložení a tlačítko "Zavřít"uzavře okno ukládání obrázků. V následujícím kusu kódu je znázorněno řešení uložení obrázku, právě když je vybrán formát obrázku png.

```
                                                    ...
else if (rbPng.Checked == true) // když je vybrán png
{
   SaveFileDialog ulozObrazek = new SaveFileDialog();
   ulozObrazek.Filter = "PNG (*.png)|*.png | Jiný (*.*)|*.*";  // uložení v png nebo jiným
   ulozObrazek.FilterIndex = 0;
   ulozObrazek.RestoreDirectory = true;
   ulozObrazek.FileName = "obrázek";  // defaultní název obrázku

   if (ulozObrazek.ShowDialog() == DialogResult.OK)  // pokud potvrdím, ulož
   {
      obr.Save(ulozObrazek.FileName, ImageFormat.Png);
      tsStatus.Text = datum + " uložen do: (" + ulozObrazek.FileName + ")";
   }
   else if (rbJpg.Checked == true)
   {
}
                                                    ...
```


Obrázek 21: Okno ukládání obrázku

# 5 Experimenty

V této části testů prověřím algoritmus a případně opravím drobné chyby v programu. V první řadě jsem otestoval funkčnost programu na několika webkamerách. U žádné z testovaných webkamer se v programu neobjevovaly problémy, pouze u několika z nich trvalo první zapnutí déle, to bylo však způsobeno pomalým startem webkamer. Testoval jsem jak webkamery interní či externí, připojené přes rozhraní USB. Dále webkamery, které se lišily kvalitou snímání, podporou zachycení obrazu i rozlišením, které dokázaly zachytit.

Další funkce, které testuji jsou tranformace a filtry, naimplementované a použité z knihoven. Důvodem je optimální výběr mezi kvalitou provedení dané trasformace či funkce a její rychlostí. Tyto testy jsou zaznamenány v dalších odstavcích.

## 5.1 Časové testy

Pro časové testy jsem si dočasně vytvořil nové tlačítko a ve zdrojovém kódu jsem importoval knihovnu pro diagnostiku. Po kliknutí na toto tlačítko se vytvoří jednoduché systémové stopky, které pro náš účel měření stačí. Mezi vytvořením, spuštěním a zastavením stopek se provede vždy jedna transformace či filtr úpravy. Jednotlivé úpravy jsem postupně měnil a měřil čas, za jaký se provedou. Ukázkové měření rychlosti transformace je na kódu pod tímto textem.

```
                                                    ...
using System.Diagnostics;
                                                    ...
private void buTest_Click(object sender, EventArgs e)
{
   Stopwatch stopky = Stopwatch.StartNew();  // Vytvoření a spuštění stopek
   bit2 = RotaceNearest(bit, 33);  //  interpolace nejbližší soused
   //bit2 = RotaceBilinear(bit, 33);
   //bit2 = RotaceBicubic(bit, 33);
   stopky.Stop();  // zastavení stopek
   MessageBox.Show(stopky.ElapsedMilliseconds.ToString() + "ms"); // zobrazení výsledného času
                                                    ...
```

Následující tabulka zobrazuje jednotlivé časy transformací rotace a změny velikosti. Měřil jsem čas při otočení obrázku o 33 stupňů při použití různých druhů interpolací, stejně tak jsem měřil zvětšení obrázku o 50 % při použití různých interpolací. Nejlepší poměr času a kvality má interpolace bilineární, kterou také budu přednostně používat v automatickém nastavení.

| **Název obrázku:**<br>**Rozlišení (px):** | **Obr1**<br>320×240 | **Obr2**<br>1280×1024 | **Obr3**<br>2592×1944 | **Obr4**<br>3264×2448 |
|---|---|---|---|---|
| **Rotace o 33°** | | | | |
| Nejbližší soused | 23ms | 121ms | 372ms | 565ms |
| Bilineární | 63ms | 114ms | 484ms | 769ms |
| Bikubická | 99ms | 994ms | 3713ms | 5912ms |
| **Zvětšení o 50 %** | | | | |
| Nejbližší soused | 12ms | 92ms | 279ms | 438ms |
| Bilineární | 56ms | 232ms | 773ms | 1204ms |
| Bikubická | 203ms | 2865ms | 10296ms | 31703ms |

Stejně jako tomu bylo u transformací jsem využil vytvořené tlačítko a stopky, jen jsem vyměnil transformace za filtr prahování a změřil čas pro různé druhy. Avšak oproti transformacím nebyly rozdíly času tak markatní, ale naproti tomu rozdíly v kvalitě byly větší. Proto jsem se rozhodl používat lokální prahování "Bradley", které má nejlepší průměrné výsledky.

| **Název obrázku:**<br>**Rozlišení (px):** | **Obr1**<br>320×240 | **Obr2**<br>1280×1024 | **Obr3**<br>2592×1944 | **Obr4**<br>3264×2448 |
|---|---|---|---|---|
| **Prahování** | | | | |
| Bradley | 56ms | 146ms | 570ms | 906ms |
| Iterative | 41ms | 136ms | 245ms | 438ms |
| Otsu | 51ms | 69ms | 142ms | 239ms |
| SIS | 59ms | 122ms | 377ms | 581ms |

## 5.2 Úspěšnostní testy

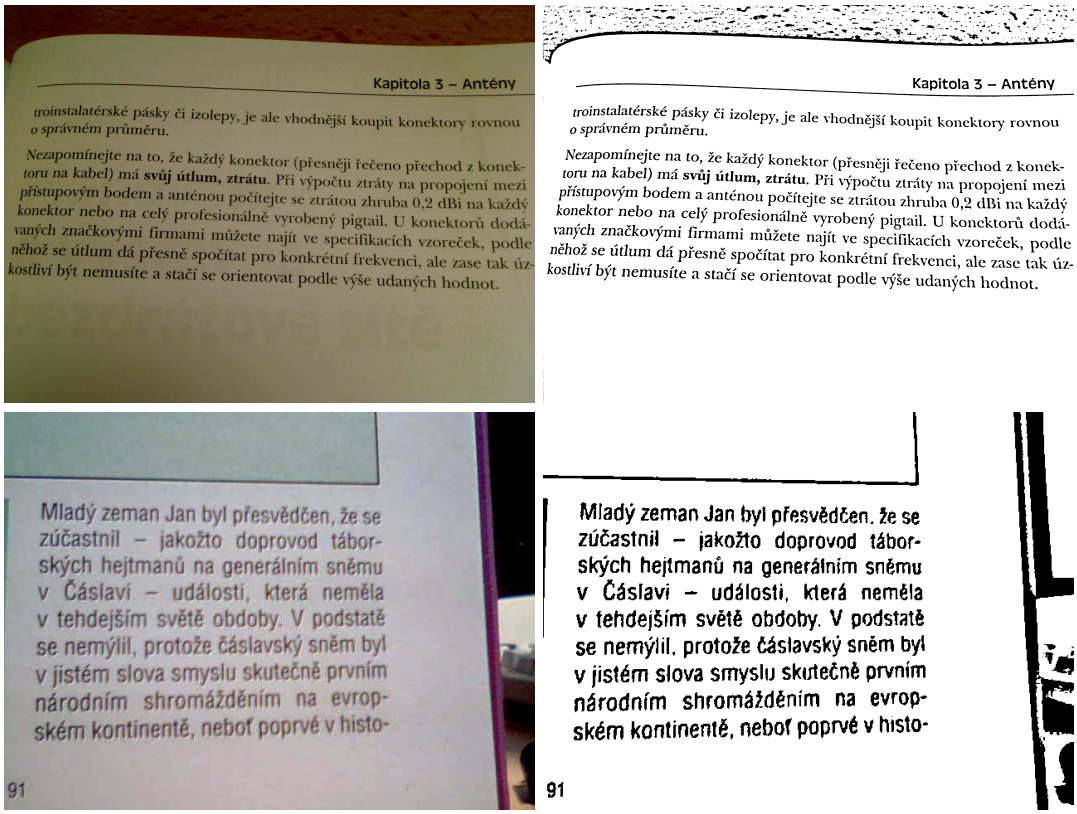
Druhou kapitolu testů jsem nazval úspěštnostní. V této fázi se budu snažit využít dostupných OCR programů pro překlad textů před úpravou a po úpravě, následně budu výsledky porovnávat v tabulce. Zvolil jsem záměrně texty o několika málo řádcích, aby bylo porovnávání výsledků jednodušší.

Obrázky jsou seřazeny od těch s největším rozlišením až po ty s rozlišením nejmenším. Využil jsem dostupné trial verze dvou desktopových aplikací a dvě verze online OCR překladačů, které podporují české znaky. Ve výsledcích překladačů budu započítávat pouze celá správná slova.

| **Originál/upravený:**<br>**Počet slov:** | **Obr1/Obr1a**<br>95 slov | **Obr2/Obr2a**<br>59 slov | **Obr3/Obr3a**<br>35 slov | **Obr4/Obr4a**<br>49 slov |
|---|---|---|---|---|
| **Použitý program** | . Správně přeložených slov u originálního / upraveného obrázku | | | |
| ABBYY FineReader 11 | 83/89 | 53/56 | 35/35 | 47/42 |
| TOCR 3.3 | 11/73 | 0/43 | 28/28 | 28/25 |
| www.onlineocr.net | 87/93 | 59/55 | 33/34 | 43/46 |
| www.free-ocr.com | 84/89 | 1/45 | 21/32 | 43/38 |

Výsledky OCR překladačů mohou být částečně zkreslené, protože nevím, jaké další filtry a transformace tyto programy při překládání používají a co na výsledném obrázku tyto úpravy změní. Navíc ve výsledcích jsou zahrnuta pouze slova, která jsou součástí textu. Některé překladače přidávaly různé znaky mezi text či do volného prostoru, které jsem nepočítal. U několika obrázků se stalo, že program použil špatnou metodu filtrování, tudíž byl obrázek téměř nečitelný a výsledkem byla směs nesmyslných znaků s několika slovy či písmeny mezi těmito znaky.

Nejhůře v testech skončil poslední obrázek (Obr. 22), který měl malé rozlišení a text nebyl tak dobře čitelný jako na ostatních obrázcích, navíc po automatické úpravě se jeho čitelnost ještě zhoršila. V tomto případě by se hodilo využít pokročilých nastavení v programu a obrázek upravit podle vlastního uvážení. U většiny obrázků s vyšším rozlišením se po automatické úpravě pro OCR programy čitelnost zlepšila a to se také promítlo do výsledků. Tudíž lze říci, že program, který jsem vytvořil, má na výsledný překlad příznivý vliv a tak je obrázek s textem po úpravě přeložen s lepšími výsledky. Ukázky dvou překládaných obrázků jsou na další stránce.

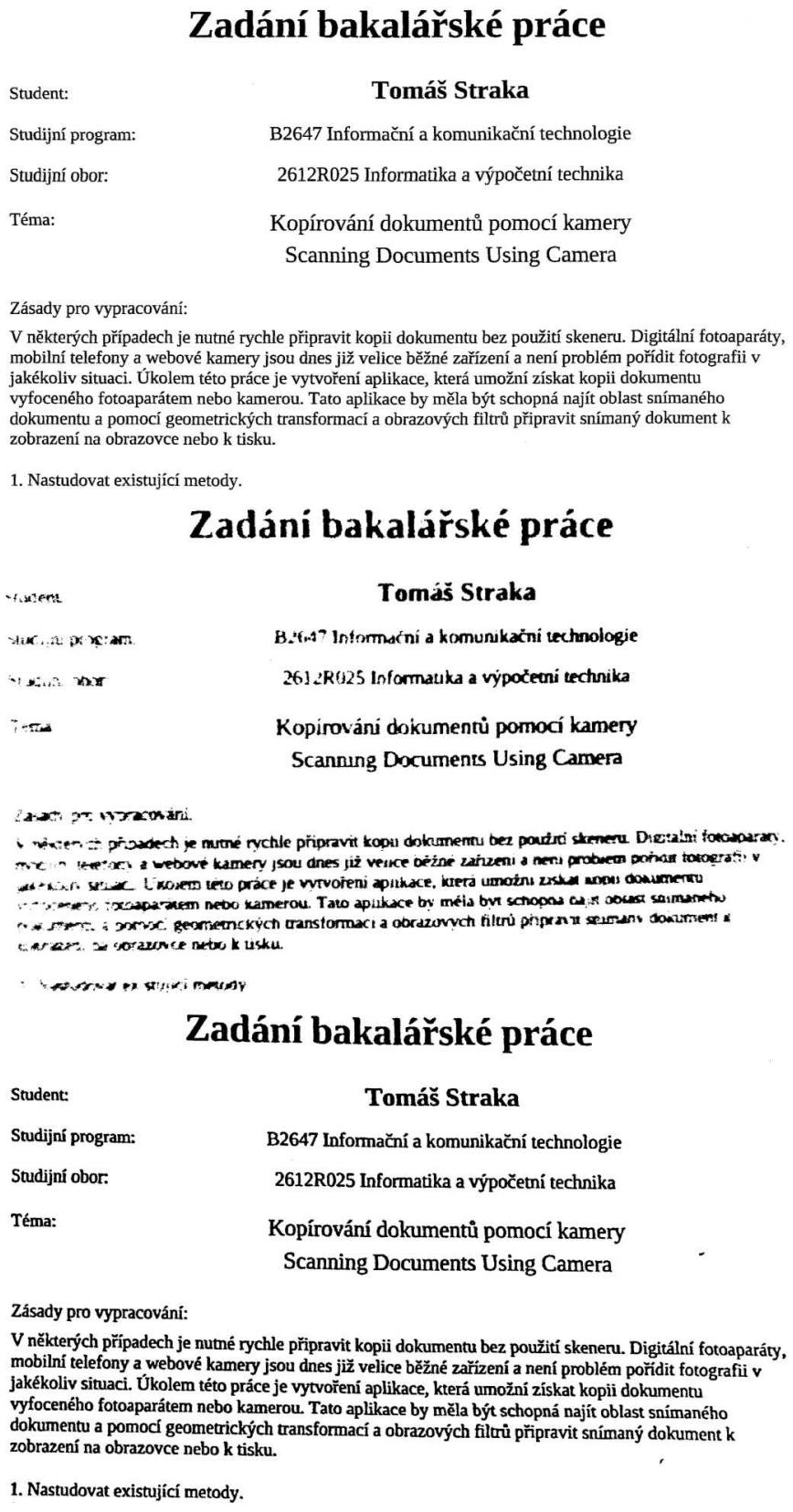



Obrázek 22: Srovnání obrázků před a po úpravě, první dva obrázky jsou obrázky Obr1 a Obr1a z tabulky experimentů překladu textu pomocí OCR a další dva jsou Obr4 a Obr4a.

Cílem závěrečného testu je ověřit funkčnost mého programu, proto jsem využil existující dokument, konkrétně zadání mé bakalářské práce ve vytištěné podobě. K testu jsem použil dvě webkamery, přičemž první je obyčejná vestavěná webkamera notebooku s průměrným rozlišením (1280×1024). Navíc obraz z webkamery nebyl příliš ostrý, od středu do stran se rozmazával a byla znát komprese obrazu. První webkamera neumožňovala vyfotit dokument, pouze vytvářet obrázky z videa.

Druhá webkamera umožňuje zachytit dokument mnohem kvalitněji bez větších neduhů, navíc umožňuje pořízení fotografií ve vysokém rozlišení (3264×2448). Část textu jsem nejdříve vyfotografoval oběma webkamerami, následně jsem pořízené fotografie upravil přes automatickou úpravu mého programu a ještě obrázky ořezal. Dalším krokem bylo vytisknutí na inkoustové tiskárně. Tento vytisknutý text jsem poté naskenoval a porovnal jednotlivé dokumenty mezi sebou.

Všechny tři dokumenty jsou pro porovnání zobrazeny na následující stránce (Obr. 23). První obrázek je naskenovaný originál, následující je zachycený webkamerou notebooku, přičemž pouze velký text je zde čitelný, menší text lze částečně přečíst pouze na středu obrázku. To je nejspíše způsobeno ostřením webkamery, jinak jednotlivá písmena splývají, nebo chybí úpně. Poslední obrázek je na tom podstatně lépe, celý text lze bez jakýchkoli problémů přečíst, oproti originálu se na textu změnily pouze drobnosti: zvětšila se tloušťka textu, na obrázku se objevují drobné šmouhy a některá písmena jsou mírně deformovaná a částečně zubatá.



# Zadání bakalářské práce

Student: **Tomáš Straka**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Kopírování dokumentů pomocí kamery
Scanning Documents Using Camera

Zásady pro vypracování:

V některých případech je nutné rychle připravit kopii dokumentu bez použití skeneru. Digitální fotoaparáty, mobilní telefony a webové kamery jsou dnes již velice běžné zařízení a není problém pořídit fotografii v jakékoliv situaci. Úkolem této práce je vytvoření aplikace, která umožní získat kopii dokumentu vyfoceného fotoaparátem nebo kamerou. Tato aplikace by měla být schopná najít oblast snímaného dokumentu a pomocí geometrických transformací a obrazových filtrů připravit snímaný dokument k zobrazení na obrazovce nebo k tisku.

1. Nastudovat existující metody.


# Zadání bakalářské práce

**Tomáš Straka**

B2647 Informační a komunikační technologie

2612R025 Informatika a výpočetní technika

Kopírování dokumentů pomocí kamery
Scanning Documents Using Camera



# Zadání bakalářské práce

Student: **Tomáš Straka**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Kopírování dokumentů pomocí kamery
Scanning Documents Using Camera

Zásady pro vypracování:

V některých případech je nutné rychle připravit kopii dokumentu bez použití skeneru. Digitální fotoaparáty, mobilní telefony a webové kamery jsou dnes již velice běžné zařízení a není problém pořídit fotografii v jakékoliv situaci. Úkolem této práce je vytvoření aplikace, která umožní získat kopii dokumentu vyfoceného fotoaparátem nebo kamerou. Tato aplikace by měla být schopná najít oblast snímaného dokumentu a pomocí geometrických transformací a obrazových filtrů připravit snímaný dokument k zobrazení na obrazovce nebo k tisku.

1. Nastudovat existující metody.


Obrázek 23: Originální text, text vyfotografovaný webkamerou notebooku, externí webkamerou

# 6 Závěr

Hlavním úkolem mé bakalářské práce bylo nastudovat používané metody a postupy řešení kopírování textu. Nastudované informace zpracovat a snažit se použít. Využil jsem běžně uplatňované postupy zpracování obrazu, vybral znich ty, které se nejlépe hodí ke kopírování textu. Tyto postupy jsem rozebral a zabýval se jimi v teoretické části této práce.

Teoretické poznatky jsem využil v praktické implementaci aplikace. Vytvořená aplikace umožnuje zachytit obraz pomocí připojené webkamery a následně upravit text do čitelnější podoby. Z programu je možné tento upravený text uložit ve formě obrázku nebo přímo vytisknout na tiskárně. Dalšími možnostmi, kterými by v budoucnu program mohl disponovat, je rozšíření o použití OCR programu a následný export čistého textu. Vhodná by byla implementace detekce hran papíru pomocí Houghovy transformace nebo paralelizování některých algoritmů v kódu.

Praktické výsledky experimentů poukazují na to, že pomocí naimplementovaného programu se zlepší čitelnost textu jak pro lidské oko, tak i pro softwarové překladače textu. Avšak to se nedá říci pro obrázky s nízkým rozlišením a horší čitelností, protože u nich dochází pomocí automatické úpravy v programu často k horším výsledkům.

Celkově si myslím, že moje bakalářská práce úspěšně uvádí do problematiky kopírování textu a otevírá další možnosti tvorby navazujících prací či programů. Především pro tvorbu identifikace textu pomocí OCR, ale i v dalších oblastech, které mohou z okruhu kopírování textu vycházet.

# Literatura


[1] Bob Fisher *"Department of Computer Science: Colour Image Processing"* The University of Edinburgh [online]. 15 May 1999 [cit. 2012-04-25]. Dostupné z: http://homepages.inf.ed.ac.uk /rbf/CVonline/LOCAL_COPIES/OWENS/LECT14/

[2] Robert Fisher, Simon Perkins, Ashley Walker and Erik Wolfart *"Thresholding"* The University of Edinburgh [online]. ©2003 [cit. 2012-04-25]. Dostupné z: http://homepages.inf.ed.ac.uk /rbf/HIPR2/threshld.htm

[3] Bob Fisher *"Spatial domain methods"* The University of Edinburgh [online]. 10/29/1997 [cit. 2012-04-25]. Dostupné z: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/ OWENS/LECT5/node3.html

[4] M. Alex O. Vasilescu *"Image Transformations: (global and local warps)"* MIT Media Lab [online]. 2008 [cit. 2012-04-25]. Dostupné z: http://web.media.mit.edu/ maov/classes/ comp_photo_vision08f/lect/08_image_warps.pdf

[5] Harvey Rhody *"Lecture 2: Geometric Image Transformations"* Chester F. Carlson Center for Imaging Science: Rochester Institute of Technology [online]. October 4, 2005 [cit. 2012-04-25]. Dostupné z: http://www.cis.rit.edu/class/simg782/lectures/lecture_02/lec782_05_02.pdf

[6] Harvey Rhody *"Lecture 17: Image Enhancement and Spatial Filtering II "* Chester F. Carlson Center for Imaging Science: Rochester Institute of Technology [online]. September 8, 2005 [cit. 2012-04-25]. Dostupné z: http://www.cis.rit.edu/class/simg782/lectures/lecture_08/ lec782_05_08.pdf

[7] Pascal Getreuer *"Linear Methods for Image Interpolation, Image Processing On Line"* [online]. 2011-09-27 [cit. 2012-04-25]. Dostupné z: http://www.ipol.im/pub/algo/ g_linear_methods_for_image_interpolation

[8] Ariel Shamir *"Seam Carving for Content-Aware Image and Video Retargeting"* Efi Arazi School of Computer Science [online]. 2007 [cit. 2012-04-25]. Dostupné z: http://www.faculty.idc.ac.il/arik/SCWeb

[9] QUADRILATERAL SIGNBOARD DETECTION AND TEXT EXTRACTION *"Angela Tam, Hua Shen, Jianzhuang Liu, and Xiaoou Tang. Multimedi Laboratory"* [online]. 2003 [cit. 2012-04-25]. Dostupné z: http://mmlab.ie.cuhk.edu.hk/2003/CISST03_Signboard.pdf

[10] Wikipedia *"Homography. In: Wikipedia: the free encyclopedia"* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-04-25]. Dostupné z: http://en.wikipedia.org/wiki/ Homography

[11] Wayne Fulton *"Brightness/Contrast and assorted Histogram Practice"* A few scanning tips: by Wayne Fulton [online]. © 1997-2002 [cit. 2012-04-25]. Dostupné z: http://www.scantips.com/bce.html

[12] Kamil Kopecký, David Nocar, Roman Kopecký *"OCR technologie v pedagogických disciplínách"* E-Pedagogium [online]. 28. 1. 2003 [cit. 2012-04-25]. Dostupné z: http://epedagog.upol.cz/eped3.2003/clanek03.htm

[13] *"AForge.NET framework"* [online]. © 2008-2011 [cit. 2012-04-25]. Dostupné z: http://aforgenet.com/framework/

## Seznam příloh

Příloha 1: CD se zdrojovými soubory

**Obsah CD:**

1. Text bakalářské práce (ve formátu pdf)
2. Spustitelný program
   (a) CopyText.exe (vlastní spustitelná aplikace)
   (b) Aforge.Net knihovny
   (c) Nápověda.help (soubor s nápovědou programu)
3. Zdrojové soubory
   (a) Zdrojové soubory programu (zdrojové soubory jazyku C#)
   (b) Zdrojové soubory textu (zdrojové soubory typografického programu Latex, obrázky...)