České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Petr Bílek

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: Automatické vyhodnocení d2 testu pomocí metod počítačového vidění

Pokyny pro vypracování:

Navrhněte a implementujte program pro automatické vyhodnocování D2 testu pozornosti. Předpokládejte pořízení obrazu D2 testu pomocí kamery na stativu nebo skeneru. Obraz vyplněného testu porovnejte s maskami správně a špatně zaškrtnutých políček uložených v souboru.
Vyhodnoťte počty odpovídajích políček na každém řádku (dle logiky vyhodnocování D2 testu). Statistiku vyplněného testu uložte do souboru vhodného pro pozdější použití v tabulkovém editoru (např. csv).
Předpokládejte potenciální použití programu jak na PC s operačním systémem Linux, tak i s MS Windows.
Program otestujte na datech dodaných zadavatelem. Zhodnoťte chybovost vámi implementovaného automatického vyhodnocování.

Seznam odborné literatury:

- Brickenkamp, R., Zillmer, E. (1998). D2 - Test of attention. Seatle: Hogrefe & Huber.
- Gabrhel, V. (2014). Test pozornosti d2: Recenze metody. TESTFÓRUM, 3(4). doi:10.5817/tf2014-4-26.
- Wilhelm Burger, Mark J. Burge., Digital Image Processing - An Algorithmic Introduction Using Java. Springer London, 2016, second edition.

Vedoucí: Ing. David Sedláček, Ph.D.

Platnost zadání: do konce zimního semestru 2018/2019

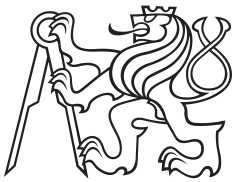
L.S.

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 4.4.2017

Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Automatické vyhodnocení d2 testu pomocí metod počítačového vidění

Bakalářská práce

Petr Bílek

Vedoucí: Ing. David Sedláček, Ph.D.
Obor: Web a multimédia
Studijní program: Softwarové technologie a management
Květen 2017

Poděkování

Chtěl bych poděkovat vedoucímu této bakalářské práce panu Ing. Davidu Sedláčkovi, Ph.D. za odborné vedení a cenné připomínky při vypracovávání této bakalářské práce.

Dále děkuji celé své rodině za podporu nejen při vypracovávání této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 26. května 2017

Abstrakt

Bakalářská práce se zabývá automatizací vyhodnocení d2 testu. Jedná se o psychologický test, který se v současné době vyhodnocuje ručně a jeho vyhodnocení zabere relativně velké množství času. Cílem bakalářské práce bylo vyvinout aplikaci, která umožní automaticky vyhodnotit formulář pro d2 test pomocí metod počítačového vidění (pomocí knihovny OpenCV) a tím usnadnit práci psychologickému pracovišti.

Klíčová slova: d2 test, počítačové vidění, OpenCV, automatizace

Vedoucí: Ing. David Sedláček, Ph.D.

Abstract

This bachelor thesis deals with automatization of evaluation of d2 test. D2 test is a psychological test which is currently evaluated manually and its evaluation takes a relatively long amount of time. The aim of the bachelor thesis was to develop an application that will allow to automatically evaluate the d2 test form using computer vision methods (OpenCV library) to facilitate the work of the psychological workplace.

Keywords: d2 test, computer vision, OpenCV, automatization

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Bakalářská práce se zabývá automatizací vyhodnocení d2 testu. Jedná se o psychologický test, který existuje v papírové podobě.
Cílem bakalářské práce bylo vytvořit aplikaci pro automatické vyhodnocování tohoto testu, která umožní vyhodnocení vyplněného formuláře pomocí metod počítačového vidění a tím usnadní práci psychologickému pracovišti, které v současné době vyhodnocování provádí manuálně.

D2 test pozornosti

V odborné literatuře nalezneme tento popis d2 testu: „*Test pozornosti d2 představuje metodu určenou k měření pozornosti, zejména pak ve smyslu pozornosti selektivní. Princip nástroje spočívá ve schopnosti probanda*[1] *rozlišovat mezi podobnými zrakovými podněty. Metodou je přitom možné měřit hned několik dílčích aspektů pozornosti. Jmenovitě jde o rychlost a celkové množství práce vyjádřené skrze počet zpracovaných písmen. Dále se jedná o chyby způsobené opomenutím správné položky (související s řízením pozornosti, s přesností zrakového rozlišování atd.) či o chyby způsobené označením nesprávných položek (související s kognitivní přizpůsobivostí či opět s přesností zrakového rozlišení).*
Skrze metodu je možné měřit hned několik dílčích aspektů (selektivní) pozornosti. Jmenovitě jde o rychlost a celkové množství práce vyjádřené skrze počet zpracovaných písmen. Dále se jedná o chyby způsobené opomenutím správné položky (související s řízením pozornosti, s přesností zrakového rozlišování atd.) či o chyby způsobené označením nesprávných položek (související s kognitivní přizpůsobivostí či opět s přesností zrakového rozlišení). Od nich se poté odvíjí další škály, např. fluktuační rozpětí (rozdíl řádků s největším a nejmenším počtem zpracovaných položek) nebo výkon soustředění (rozdíl mezi správně a špatně označenými položkami).“ [5]

[1] jedinec, který je předmětem zkoumání

Kapitola 2

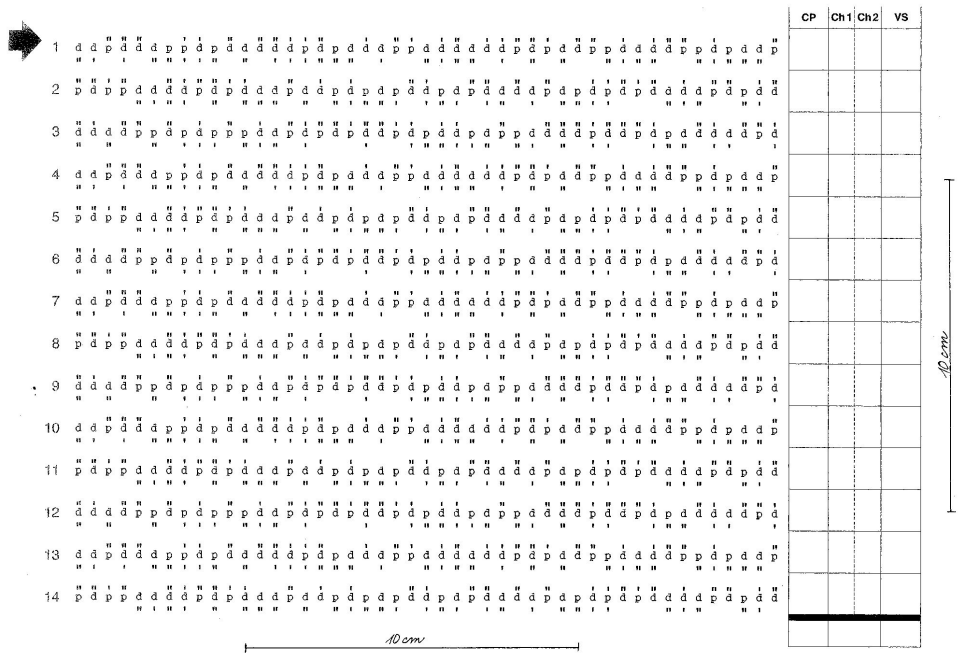
Analýza

V této kapitole je popsána současná podoba d2 testu a jeho vyhodnocování. Dále se kapitola zabývá analýzou počítačového vidění, které bude využito pro budoucí vyhodnocování testu.

2.1 Současná podoba testu a jeho vyhodnocování

Formulář d2 testu se skládá ze 14 řádků po 47 znacích. Jedná se o písmena *d* a *p* s jednou, dvěma, třemi nebo čtyřmi čárkami. Pro každé písmeno existuje šest variant umístění čárek.

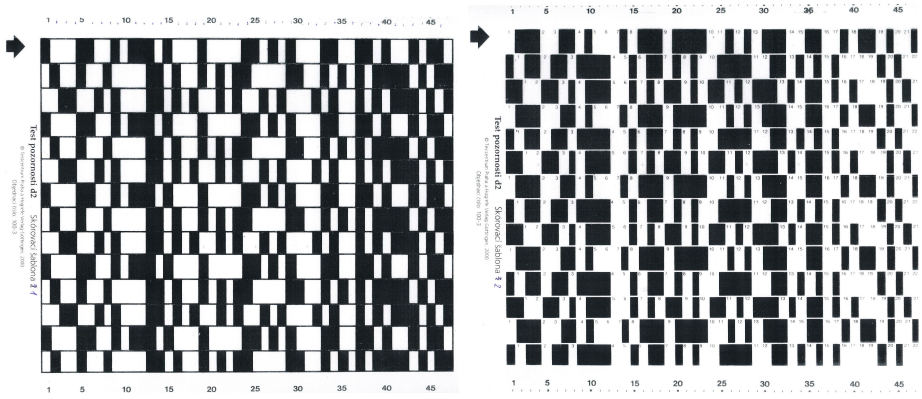
Formulář pro d2 test má následující podobu:



Obrázek 2.1: Formulář d2 testu

Testovaná osoba má během časově limitované zkoušky (20 sekund na řádek) vybrat (seškrtat) všechny vyhovující položky (písmena *d* se dvěma čárkami), přičemž ty nevyhovující má ponechat neoznačené. Testovaný se může i opravit, a to výraznějším přeškrtáním své původní volby. [5]

V současné době probíhá vyhodnocování manuálně. Na vyplněný formulář jsou přiloženy postupně dvě průhledné šablony, pomocí nichž dojde k vyhodnocení. Výsledky jednotlivých řádků se zapisují do tabulky která je umístěna v pravé časti formuláře.



Obrázek 2.2: Šablony pro vyhodnocení d2 testu

Při samotném vyhodnocování testu se zjišťuje:

- počet správně zaškrtnutých polí
- počet zaškrtnutých polí, která neměla být zaškrtnuta
- počet nezaškrtnutých polí, která měla být zaškrtnuta
- pozice posledního zaškrtnutí

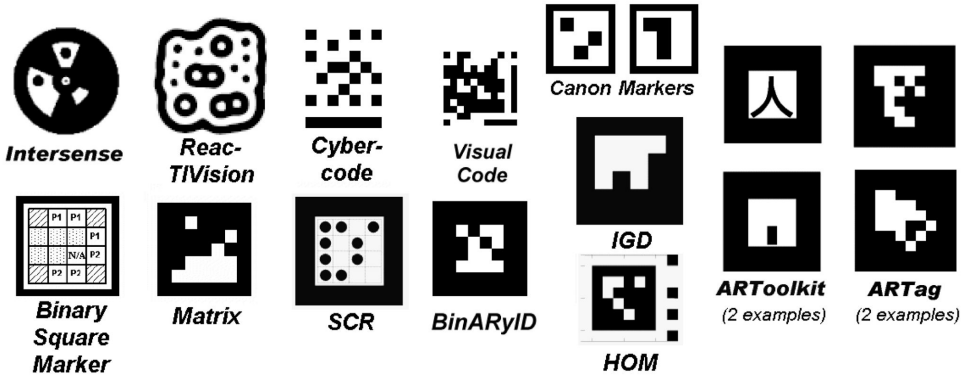
Nad výsledky je poté prováděna detailnější analýza, kterou provádí pracovníci psychologického pracoviště. Tato detailní analýza není předmětem této práce. Více informací v odborné literatuře *Vít Gabrhel, Test pozornosti d2: Recenze metody*, kapitola *Administrace a skórování* [5].

2.2 Vyhodnocování pomocí počítačového vidění

Pro budoucí vyhodnocování testu pomocí počítačového vidění je současná podoba formuláře nevyhovující. Chybí na ní záměrné body, které se v počítačovém vidění používají k zaměření obrazu.

Záměrné body

Záměrné body (markery) se skládají ze sad vzorků, které lze detekovat v obraze pomocí vhodných detekčních algoritmů. Nejčastěji mají kruhový nebo čtvercový tvar, protože tyto obrazce jsou v obraze snadno detekovatelné. [6]



Obrázek 2.3: Ukázka různých typů záměrných bodů. [24]

Více o návrhu nového formuláře v kapitole *Návrh a implementace*, podkapitole *Nový formulář* (3.3).

U zaměřeného obrazu bude potřeba provést transformaci obrazu do podoby, která bude vyhovovat strojovému zpracování a vyhodnocení.
Bude potřeba také vymyslet způsob, jakým zachycený formulář vyhodnotit.

Zejména u pořizování obrazu pomocí webové kamery nastane problém s korekcí jasu a kontrastu. Dalším problémem, který bude třeba uvažovat a ošetřit ho, jsou optické vady webových kamer.

Na řešení těchto problémů se zaměřuje kapitola *Návrh a implementace*, podkapitola *Algoritmy* (3.4).

Kapitola 3

Návrh a implementace

3.1 Požadavky na aplikaci

Jak již bylo řečeno v úvodu, cílem této práce bylo navrhnout a implementovat aplikaci pro automatické vyhodnocování d2 testu pozornosti.
Předpokládají se dva způsoby pořízení obrazu, a to pomocí kamery na stativu nebo skeneru.
Obraz vyplněného testu má být porovnán s maskami správně a špatně zaškrtnutých políček uložených v souboru a počet odpovídajících políček na každém řádku má být vyhodnocen dle logiky d2 testu (viz 2.1).
Statistika vyplněného testu má být poté uložena do souboru vhodného pro pozdější použití v tabulkovém editoru (např. csv).
Dalším předpokladem je potenciální použití programu jak na operačním systému MS Windows, tak i Linux.

Následující podkapitoly popisují návrh řešení včetně popisu použitých algoritmů. Popsána je také implementace aplikace.

3.2 Knihovna OpenCV

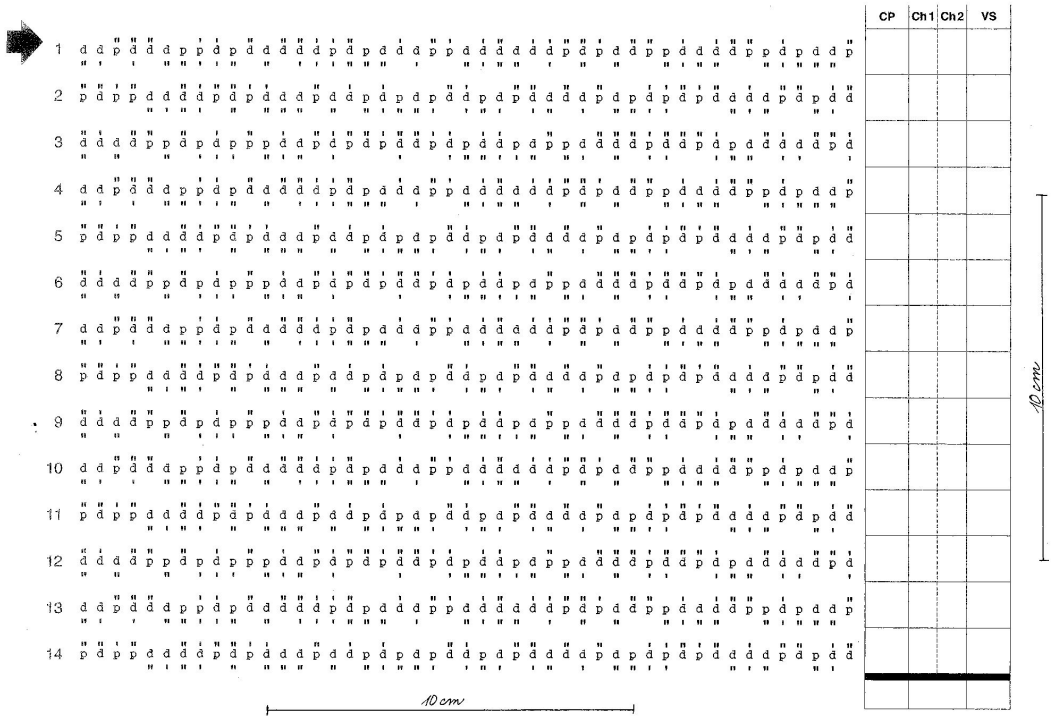
Pro potřeby aplikace byla zvolena knihovna OpenCV.
OpenCV (Open Source Computer Vision) je volně dostupná knihovna pro počítačové vidění, zpracování obrazu v reálném čase a strojové učení.
Knihovna je šířena pod BSD licencí[1], a tak je možné ji využít ke studijním i komerčním účelům. OpenCV má C++, C, Python, MATLAB a Java rozhraní a podporuje systémy Windows, Linux, iOS, Mac OS a Android.
Knihovna má více než 2500 optimalizovaných algoritmů, které zahrnují rozsáhlou sadu klasických i nejmodernějších algoritmů pro počítačové vidění a strojové učení. Tyto algoritmy mohou být použity k detekci a rozpoznání obličeje, identifikaci objektů, klasifikaci lidských akcí ve videu, sledování pohybů

[1] **Berkeley Software Distribution** licence pro svobodný software; Umožňuje volné šíření licencovaného obsahu, přičemž vyžaduje pouze uvedení autora a informace o licenci, spolu s upozorněním na zřeknutí se odpovědnosti za dílo.
Licence OpenCV: `http://opencv.org/license.html`

kamery, sledování pohybujících se objektů, extrahování 3D modelů objektů, vytváření 3D bodových mračen ze stereoskopických kamer, atd. Komunita uživatelů knihovny OpenCV dosahuje více než 47 tisíc uživatelů a počet stažení knihovny překračuje počet 14 milionů. [11]

3.3 Nový formulář

Pro potřeby aplikace musel být vytvořen nový formulář. Stará podoba byla nevyhovující pro zpracování pomocí počítačového vidění. Navíc obsahovala prvky, které pro automatické zpracování již nejsou potřeba.

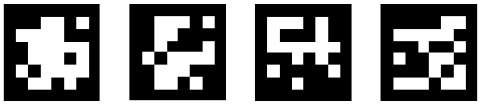


Obrázek 3.1: Stará podoba formuláře

- **Byla odebrána tabulka pro zapisování výsledků při ručním vyhodnocování.**
  Na starém formuláři se tato tabulka nacházela v pravé části formuláře. Při automatickém vyhodnocování ztrácí tato tabulka svoji funkci, a proto mohla být odebrána.

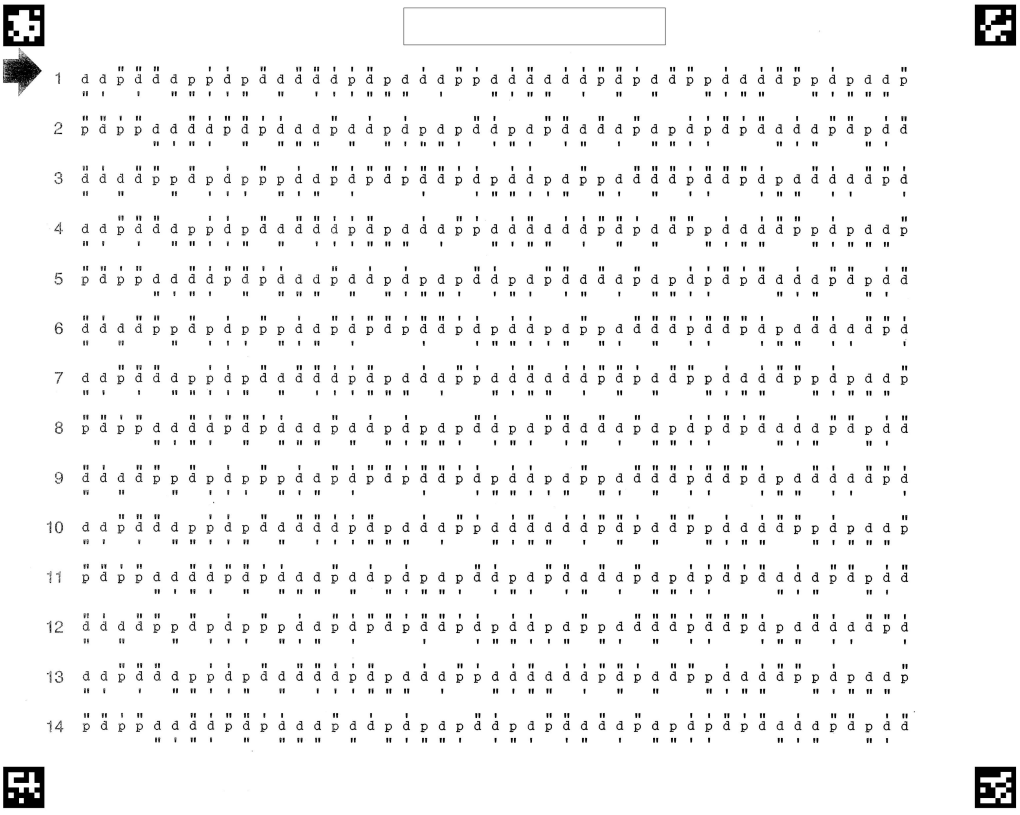
- **Bylo přidáno pole pro identifikační číslo.**
  Aby bylo možné testy identifikovat a snadněji například třídit skenované snímky, bylo pro usnadnění přidáno pole pro identifikaci účastníka.

- **Byly přidány záměrné body. (tzv. ChArUco Diamond) [31]**
  V rámci testování byly vyzkoušeny různé varianty záměrných bodů. V testech nejlépe dopadla varianta *ChArUco Diamond*, a proto byla zvolena. Mezi dalšími testovanými typy byly například záměrné body *Intersense* nebo *Reactivision*. (viz 2.2)



Obrázek 3.2: Záměrné body, tzv. ChArUco Diamond [31]

Výsledná podoba nového formuláře vypadá takto:



Obrázek 3.3: Nový formulář

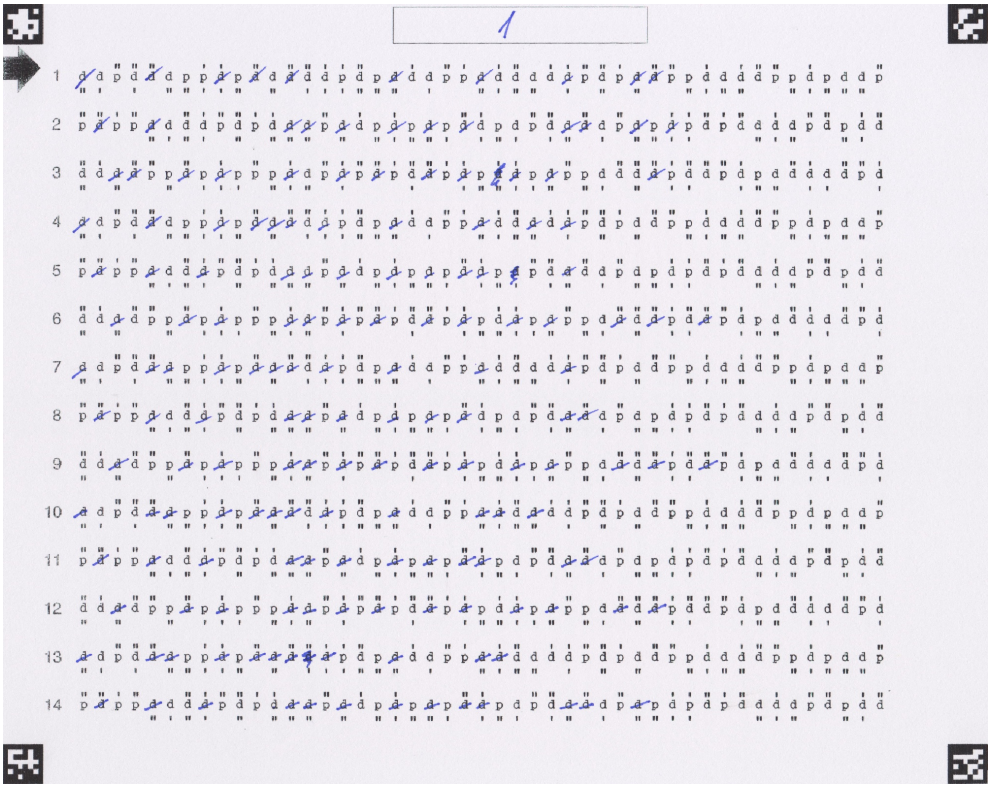
3.4 Algoritmy

V následujících podkapitolách jsou popsány jednotlivé algoritmy použité při implementaci aplikace. V převážné většině se jedná o algoritmy implementované v knihovně OpenCV (viz 3.2).

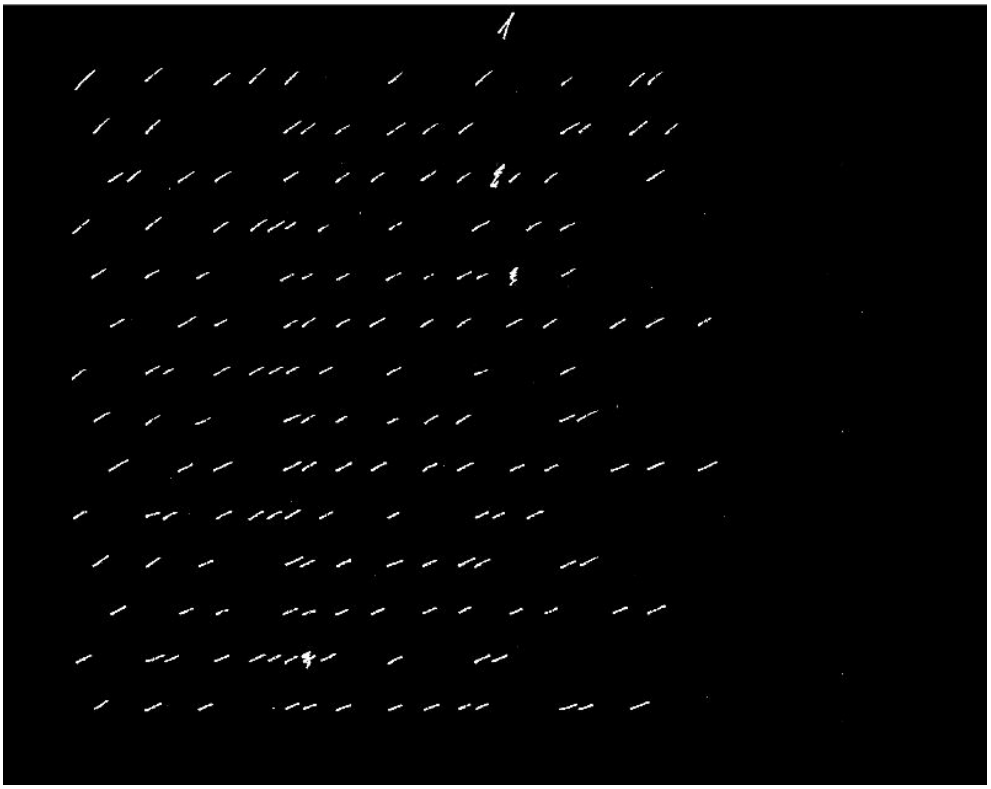
3.4.1 Detekce barvy v obraze

Základním algoritmem, jak získat výsledky zachyceného formuláře d2 testu, je detekce zaškrtnutí v obraze. To získáme pomocí detekce barvy v obraze. Ta se provádí funkcí, která omezí barevný rozsah na požadovaný interval. V programu je použita funkce *inRange* z knihovny OpenCV. (viz [19])

V aplikaci je požadovaný interval specifikován v konfiguračním souboru *config.ini* (příloha D) zvlášť pro modrou a červenou barvu. Na následujících obrázcích je vidět použití v aplikaci (*PROGRAM2*).



Obrázek 3.4: Originální obraz zaškrtaného formuláře testu d2



Obrázek 3.5: Ukázka detekce barvy (modré) v HSV (3.4.2) obrazu. Jsou vidět pixely, které odpovídají požadovanému barevnému rozsahu.

3.4.2 Převod BGR na HSV

Pro lepší práci s barvami je v programu použit převod mezi barevnými prostory. Konkrétně se používá převod mezi prostorem BGR (OpenCV používá prostor BGR namísto RGB) a HSV.

RGB prostor

„Barevný prostor RGB odpovídá fyziologii vnímání lidského oka, které má na tyto tři základní barvy speciální receptory na sítnici oka – tzv. čípky. Tyto tři základní barvy spektra lze snímat a zobrazovat v různé intenzitě. Jejich kombinace pak utvářejí všechny ostatní barevné odstíny, od černé (nulová intenzita všech tří barev) až po bílou (maximální intenzita všech tří barev).“ [4]

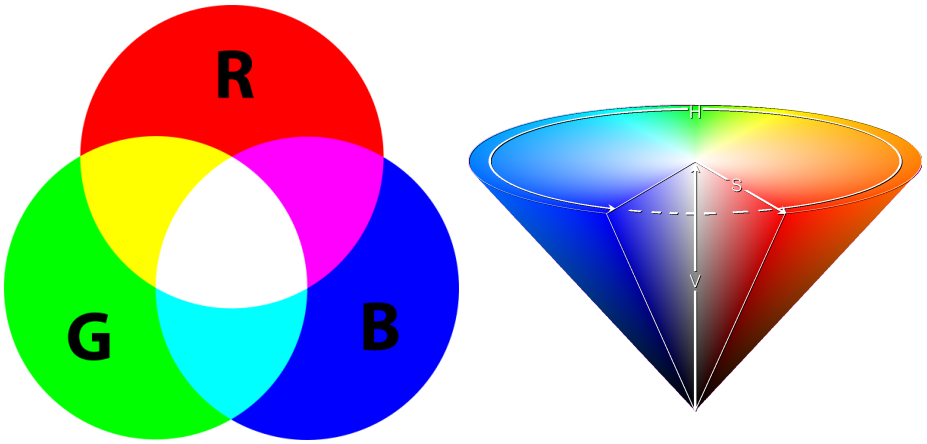
HSV prostor

V barevném modelu HSV se nové odstíny vytvářejí přidáváním bílé (vznikají nádechy) nebo černé (vznikají odstíny) do základních spektrálních barev. Prostor definuje barvu pomocí trojice složek, které však nepředstavují základní barvy.

Základními parametry jsou:

- barevný tón (H - hue) - reprezentuje převládající spektrální barvu v rozsahu od 0 do 360 stupňů (v OpenCV rozsah 0-180)
- sytost (S - saturation) - udává "čistotu" barvy a bývá vyjádřena poměrem čisté barvy a bílé
- jas (V - value) - vyjadřuje intenzitu barvy a bývá vyjádřen poměrem čisté barvy a černé

[4]
Díky těmto vlastnostem prostoru HSV lze oddělit jasovou složku, se kterou se v případě metod počítačového vidění často pracuje (vyrovnání na základě histogramu atd.).



Obrázek 3.6: Barevné prostory RGB (vlevo) a HSV (vpravo) [27, 28]

3.4.3 Sejmutí formuláře

Dále bylo potřeba navrhnout algoritmus pro sejmutí formuláře. Podstatou sejmutí je zjistit, která pole jsou zaškrtnutá, a která nikoli.
Vlastní algoritmus sejmutí formuláře je založen na "rozsekání" na jednotlivá pole, u kterých pak dochází k vyhodnocení, jestli je pole zaškrtnuté.

Postup algoritmu je následující:

- **zpracování po řádcích**
  Algoritmus "rozseká" formulář na jednotlivé řádky, které jeden po druhém zpracovává.
- **získání jednotlivých polí**
  Algoritmus "rozseká" řádek na jednotlivá pole, která uloží do dvourozměrného *vectoru*[2].

[2]datový kontejner C++ ze standardní knihovny *(std::vector)*

- **vyhodnocení zaškrtnutí na základě parametrů**
  Algoritmus vyhodnotí zaškrtnutí pole na základě parametrů pro minimální a maximální citlivost detekce. Pole s počtem nenulových pixelů ohraničeným těmito dvěma parametry budou vyhodnocena jako zaškrtnutá.
- **uložení výsledků do vnitřní proměnné**
  Algoritmus uloží vyhodnocení do dvourozměrného *vectoru*.

3.4.4 Vyhodnocení testu

Posledním algoritmem, který bylo potřeba navrhnout pro vyhodnocení, je algoritmus, který vyhodnotí sejmutý test.
Vlastní algoritmus vyhodnocení testu je založen na porovnání výsledku sejmutí formuláře s referenční maskou správných a špatných odpovědí.

Postup algoritmu je následující:

- **zpracování po řádcích**
  Algoritmus prochází *vector* výsledku sejmutí formuláře.
- **pro každý řádek zjištění**:
  - počtu správně zaškrtnutých polí
  - počtu zaškrtnutých polí, která neměla být zaškrtnuta
  - počtu nezaškrtnutých polí, která měla být zaškrtnuta
  - indexu posledního zaškrtnutí
- **uložení statistiky do formátu CSV**
  Ukázka výstupního souboru ve formátu CSV se nachází v příloze G.

3.4.5 Zaměření obrazu

Jelikož zachycený obraz, ať už ze skeneru nebo webové kamery, nevyhovuje zpracování pomocí počítačového vidění, je potřeba ho nejprve upravit do podoby, která bude strojovému zpracování vyhovovat. Nejprve je potřeba zaměřit ve zpracovávaném obrazu obraz referenční.
První fází zaměření je nalezení tzv. klíčových bodů a vypočítání jejich deskriptorů. K tomu slouží algoritmus SURF.

SURF

SURF (Speeded-Up Robust Features) je metoda, která dokáže popsat obrázek pomocí deskriptorů. Popis pomocí deskriptorů vygenerovaných metodou SURF je invariantní vůči rotaci a vzdálenosti kamery od popisovaného objektu.
Algoritmus SURF se využívá v mnoha aplikacích počítačového vidění. Je používán například pro rekonstrukci 2D a 3D scén, klasifikaci obrázků a

především pro rychlý popis obsahu obrázku.
Průběh metody SURF lze rozdělit na dvě fáze.
První fáze představuje nalezení klíčových bodů obrazu, kterými mohou být rohy, skvrny nebo T-spoje.
Ve druhé fázi dochází k výpočtu deskriptoru z okolí klíčového bodu. [8, 9, 13]

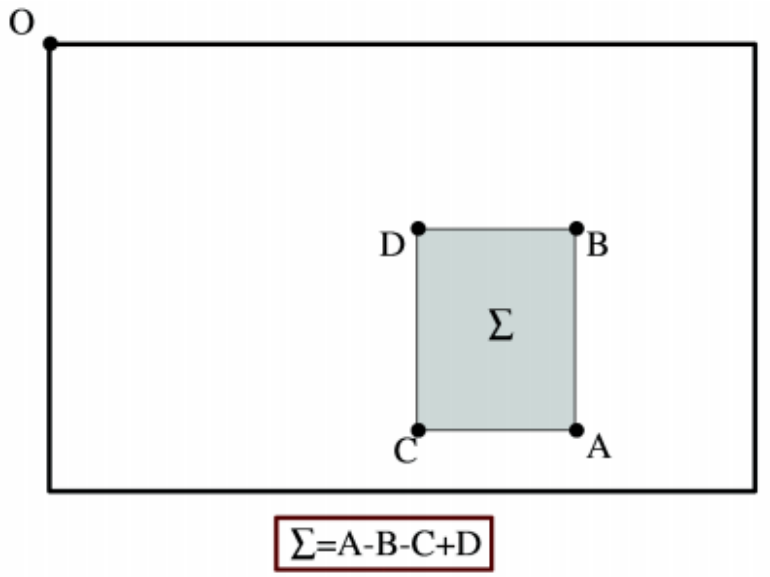
V následujícím textu budou jednotlivé fáze popsány.

Integrální obraz

Integrální obraz je způsob digitální reprezentace obrazu, kdy každý bod představuje součet hodnot předchozích pixelů od počátku obrazu. Máme-li obraz I, můžeme hodnotu bodu integrálního I_Σ obrazu vyjádřit takto:

$$I_\Sigma(x, y) = \sum_{x}^{i=1} \sum_{y}^{j=1} I(i, j) \tag{3.1}$$

Pokud máme k dispozici integrální obraz, výrazně se snižuje výpočetní náročnost, protože výpočet libovolně velké sumy přes hodnoty původního obrazu vždy představuje pouze sečtení 4 čísel. (viz obrázek 3.7) [8, 10]



Obrázek 3.7: Výpočet hodnoty plochy Σ vymezené vrcholy A, B, C a D v integrálním obraze. [29]

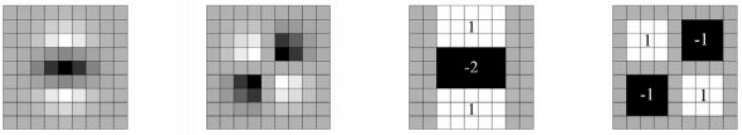
Detekce klíčových bodů

Algoritmus SURF využívá pro detekci klíčových bodů integrální obraz. (viz 3.4.5)

K detekci významných bodů v obraze se využívá detektoru založeného na výpočtu determinantu Hessovy matice. „Hessova matice má tvar:

$$H(x,\sigma) = \begin{pmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{pmatrix} \tag{3.2}$$

kde x značí souřadnice bodu obrazu o měřítku σ. $L_{xx}(x,\sigma)$ je pak konvoluce druhé derivace Gaussovy funkce s originálním obrazem I v bodě x, stejně tak $L_{xy}(x,\sigma)$ a $L_{yy}(x,\sigma)$.
Gaussova funkce je optimální funkcí pro generování scale space[3]. Z důvodu zjednodušení výpočtů zde není scale space generováno postupnými konvolucemi Gaussianu s originálním obrazem, ale jsou zde použity aproximace přímo jeho druhých derivací použitých v rovnici 3.2, což opět přispívá ke zrychlení výpočtu.“ [9, s. 16]



Obrázek 3.8: Diskretizované derivace Gaussovy funkce a jejich aproximace. Zleva druhá derivace Gaussovy funkce podle y, druhá derivace Gaussovy funkce podle xy a jejich aproximace. Šedé plochy jsou rovny nule. [9]

„Tyto aproximace uvedené v obrázku 3.8 jsou rozměru 9x9 a odpovídají nejmenšímu měřítku (tzn. největšímu prostorovému rozlišení). Další měřítka jsou pak generována postupným zvětšováním konvoluční masky, která se aplikuje stále na originální integrální obraz. Tímto způsobem jsou spočítány všechny prvky Hessovy matice ve všech měřítkách.(...) Po vypočtení determinantu jsou pak významné body hledány způsobem, že bod, který je maximem ve svém 3x3x3 okolí, je označen jako významný.“ [9, s. 17]

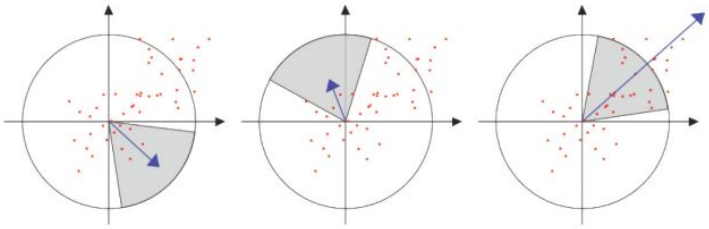
Výpočet deskriptorů

„Deskriptor popsaný specifikací algoritmu SURF představuje 64-rozměrný vektor hodnot, spočtený na okolí detekovaného významného bodu. Velikost tohoto okolí závisí na měřítku, na kterém byl daný bod detekován. Orientace tohoto okolí je určena před vlastním procesem výpočtu deskriptoru, čímž je dosaženo invariance vůči rotaci.“ [8, s. 11]

Orientace bodů jsou opět počítány pomocí konvoluce, aby se využilo integrálního obrazu a zvýšila se tak rychlost výpočtu. Jako konvoluční masky se používají Haarovy vlnky ve směru x a y. [9]

[3] spojitá funkce měřítka [3]

Pro rozhodnutí o orientaci se okolo každého významného bodu zvažuje kruhová oblast s poloměrem $6s$ (s znamená měřítko, ve kterém byl bod detekován). V této oblasti jsou vyčísleny odezvy Haarových vlnkových filtrů ve směrech os x a y. Odezvy vlnkových filtrů v dané oblasti jsou váhovány Gaussovou funkcí. [1]
„V každém bodě kruhové oblasti tedy získáme váhovanou odezvu vlnkových filtrů ve směrech x a y. Tyto dvě odezvy prohlásíme za vektor. Okolo středu stanoveného kruhu orotujeme kruhovou výseč velikosti $\frac{\Pi}{3}$, ve které sčítáme Euklidovské normy vektorů.(...) Úhel vektoru, daného součtem odezev filtrů ve výseči, která vykazuje největší součet obsažených vektorů, je poté prohlášen za dominantní a deskriptor je dále počítán ve směru tohoto úhlu.“ [8, s. 11]

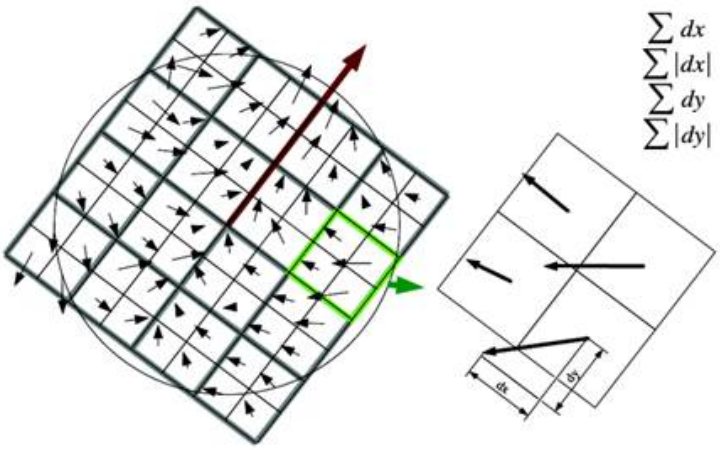


Obrázek 3.9: Princip přiřazování dominantního směru orientace deskriptoru [8]

Pro formování deskriptoru se pak zvolí čtvercová oblast kolem významného bodu. Čtverec je natočen podle orientace významného bodu a jeho velikost je přizpůsobena měřítku, ve kterém se významný bod nachází. Tato čtvercová oblast je pak rozdělena na 16 (4x4) stejně velkých čtvercových podoblastí. Každá z nich obsahuje 25 (5x5) rovnoměrně rozložených vzorkovacích bodů. V každém z nich jsou spočteny odezvy Haarových vlnkových filtrů pro směry d_x a d_y. Zmíněné směry d_x a d_y ale nereprezentují směry obrazových os x a y. Směr d_x představuje směr odpovídající orientaci deskriptoru určený dle odstavce výše a směr d_y je směr na něj kolmý.
Každá z 16 podoblastí je poté popsána 4-rozměrným vektorem:

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \tag{3.3}$$

Tyto vektory poskytují 4x4x4 = 64 hodnot, které tvoří základ SURF deskriptoru. [1, 8, 9]



Obrázek 3.10: Postup výpočtu SURF deskriptoru [25]

Použití SURF v aplikaci

V aplikaci (*PROGRAM1*) je algoritmus SURF využit prostřednictvím knihovny OpenCV. (viz [22])

FLANN matcher

Další fází zaměření obrazu je napárování deskriptorů referenčního a zpracovávaného obrazu.
K tomu slouží algoritmus **FLANN (Fast Library for Approximate Nearest Neighbors)**, který provádí rychlé a účinné párování shod nad kolekcí deskriptorů.

Jedná se o algoritmus, jehož použití je možné prostřednictvím knihovny OpenCV. (viz [16])
V aplikaci je FLANN matcher použit v *PROGRAMu1* k párování shod deskriptorů referenčního obrazu a obrazu ke zpracování.

Obrázek 3.11: Ukázka párování shod deskriptorů pomocí algoritmu FLANN matcher. [26]

3.4.6 Transformace obrazu

Zaměřený obraz z předchozí kapitoly je nyní potřeba transformovat do referenčního obrazu.
Nejprve je potřeba nalézt matici transformace a pomocí ní poté obraz transformovat. K prvnímu kroku se použije funkce *findHomography*, ke druhému pak funkce *warpPerspective*, obě z knihovny OpenCV.

findHomography

Funkce slouží k nalezení perspektivní transformace mezi dvěma rovinami.

Jako vstup slouží pole bodů pro obě roviny. Pole obsahují pozice významných bodů v obrazech. V obou polích jsou body uspořádány tak, že body na stejných indexech jsou body korespondující. Funkce *findHomography* se snaží vytvořit homografii tak, aby po aplikaci matice homografie na pozice bodů v poli referenčního obrazu souhlasily výsledné pozice s pozicemi bodů v poli zpracovávaného obrazu na stejných indexech. Jedním z parametrů funkce je i metoda hledání klíčových bodů. V implementaci je použita metoda RANSAC. [2, 15]

RANSAC hledá matici homografie takovou, aby chyba zpětné projekce (back-projection error) byla co nejmenší. „RANSAC v každé iteraci vybere náhodně čtyři korespondence, pro ty se vypočítá matice homografie a ohodnotí se všechny korespondence např. pomocí Symmetric transfer error (také nazýváno back-projection error) 3.4, což je metoda, která hodnotí, jak se transformace povedla.

$$d^2_{transfer} = d(x, H^{-1}x')^2 + d(x', Hx)^2 \tag{3.4}$$

, kde x a x' jsou korespondenční body, H je homografie z prostoru bodu x do prostoru bodu x' a d je funkce výpočtu euklidovské vzdálenosti bodů.
Podle této hodnoty se určí úspěšnost dané matice homografie a pro finální použití se algoritmem RANSAC vybere ta, která se při běhu jevila jako nejlepší (nejmenší Symmetric transfer error).“ [2, s. 23]

warpPerspective

Funkce transformuje obraz pomocí následující rovnice (pokud je nastaven flag *WARP_INVERSE_MAP*, který nastaví M jako inverzní matici (`dst` $\rightarrow$ `src`)):

$$\texttt{dst}(x, y) = \texttt{src}\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right) \tag{3.5}$$

kde M je transformační matice 3x3, $\texttt{src}(x, y)$ jsou souřadnice bodu ve vstupním obraze a $\texttt{dst}(x, y)$ jsou souřadnice bodu ve výstupním obraze.(viz [23])

3.4.7 Kalibrace kamery

Pokud se pro zachycení obrazu použije varianta webové kamery, je potřeba nejprve provést její kalibraci. Procesem kalibrace kamery se nazývá nalezení **matice kamery** a **matice s parametry zkreslení**.

Parametry **matice kamery** vycházejí z rovnice:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.6}$$

kde $(X, Y, Z)^T$ jsou souřadnice bodu v prostoru a $(x, y, w)^T$ jsou souřadnice bodu v projekci. Přítomnost w je vysvětlena použitím homografického souřadného systému (a $w = Z$). Neznámé parametry jsou f_x a f_y (ohnisková vzdálenost kamery) a (c_x, c_y), což jsou optická centra vyjádřená v souřadnicích pixelů. Pokud pro obě osy používáme společnou ohniskovou vzdálenost s daným poměrem stran α (obvykle 1), pak $f_y = f_x * \alpha$ a v horním vzorci budeme mít jednu ohniskovou vzdálenost f. Právě matice obsahující tyto čtyři parametry je označována jako matice kamery. [14]

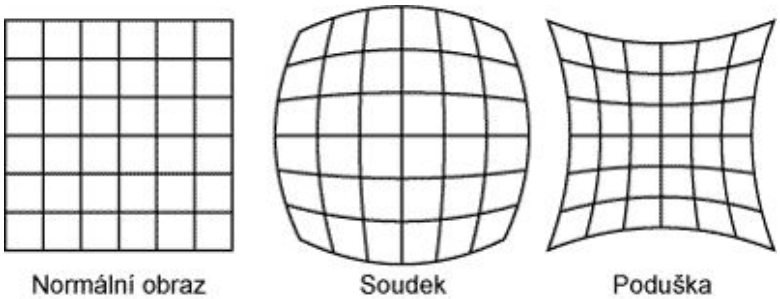
V OpenCV se jedná o matici 3x3 ve tvaru:

$$\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.7}$$

Parametry matice kamery se s různým rozlišením mění. [14]

Dále je potřeba nalézt **matici s parametry zkreslení**. Hlavní roli hrají dva typy zkreslení - radiální a tangenciální.

Radiální zkreslení se projevuje buď zakřivením obrazu do koule (soudek, Barrel) nebo prohnutím obrazu (poduška, Pincushion).



Obrázek 3.12: Typy radiálního zkreslení [27]

Radiální zkreslení je typické pro širokoúhlé objektivy nebo pro zoom objektivy na širokoúhlém konci. Je dobře viditelné tam, kde jsou na snímku výrazné a rovné čáry blízko okraje (šachovnice, rám okna, horizont moře). [12]
„Příčinou radiálního zkreslení je tvar čočky v objektivu kamery. Viditelně se projevuje tím, že přímky ve scéně se v obraze nekalibrované kamery zobrazují jako oblé křivky. Soudkovitost, naznačená na 3.12, je vlastnost objektivu posouvat vnímané obrazové body směrem ke středu obrazu. Naopak tzv. poduškovitost, obrázek 3.12, je tendence k posunu obrazů směrem k okrajům obrazu. (...) Původní bod X_c v souřadnicích kamery je působením radiálního zkreslení posunut do bodu X_d podle vztahu:

$$\begin{bmatrix} x \\ y \end{bmatrix} = Z(r) \begin{bmatrix} x_c \\ y_c \end{bmatrix} \tag{3.8}$$

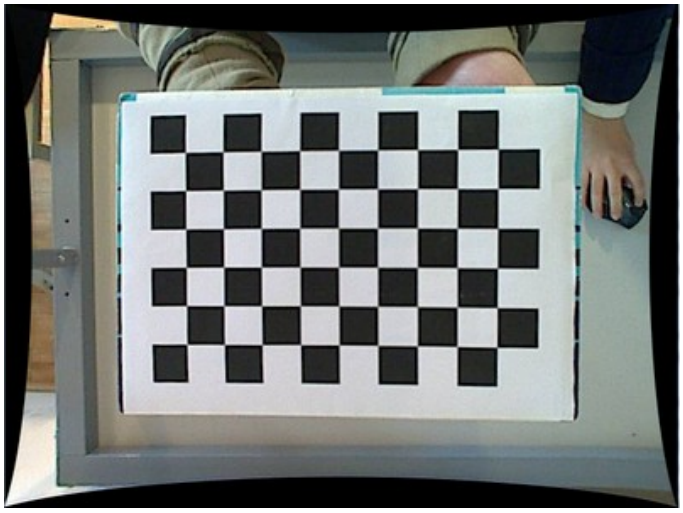
kde $(x_c, y_c)^T$ značí nezkreslené souřadnice a $(x, y)^T$ souřadnice pozorované (zkreslené). Funkce Z v proměnné $r = \sqrt{x_c^2 + y_c^2}$ vyjadřuje zkreslení závislé na vzdálenosti od středu radiálního zkreslení, které nemusí nutně ležet v hlavním bodě kamery. Funkce $Z(r)$ bývá nejčastěji aproximována Taylorovým polynomem. Protože jde o funkci sudou, jsou v rozvoji zastoupeny pouze sudé mocniny r.“ [7, s. 8] Absolutní člen a_0 je podle rovnice 3.8 roven jedné:

$$Z(r) = a_0 + a_2 r^2 + a_3 r^4 = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \tag{3.9}$$

[14]
„Obraz je možné opravit podle rovnic 3.10. Stanovení koeficientů k_i je součástí stanovení vnitřních parametrů kamery a lze jej provést pomocí identifikace odchylek zkreslených přímkových segmentů.“ [7, s. 8]
Ke zjištění těchto koeficientů se používají kalibrační vzory (např. obrázek 3.13). „Pokud kalibrační vzor obsahuje dostatečné množství bodů ležících na přímce, lze z odchylek těchto bodů od přímky procházející dvěma krajními body koeficienty radiálního zkreslení stanovit.“[s. 8] [7]

$$\begin{aligned} x_c &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_c &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{3.10}$$

Pro staré souřadnice (zkreslené) bodu ve vstupním obraze (x, y) jsou opravené souřadnice (nezkreslené) (x_c, y_c). [14]



Obrázek 3.13: Jako kalibrační vzor se často používá šachovnice [14]

Dalším typem zkreslení je zkreslení tangenciální. To vzniká v důsledku tolerance při výrobě kamer, kdy optický senzor není v těle kamery umístěn rovnoběžně s optickou rovinou objektivu.
Tangenciální zkreslení se vyjadřuje dvěma parametry p_1, p_2 a korekce je provedena podle rovnic 3.11. Obvykle se tangenciální zkreslení projevuje v menší míře než zkreslení radiální. [7]

$$\begin{aligned} x_c &= x + [2p_1xy + p_2(r^2 + 2x^2)] \\ y_c &= y + [p_1(r^2 + 2y^2) + 2p_2xy] \end{aligned} \tag{3.11}$$

Parametry zkreslení reprezentují v OpenCV prvky matice:

$$\begin{pmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{pmatrix} \tag{3.12}$$

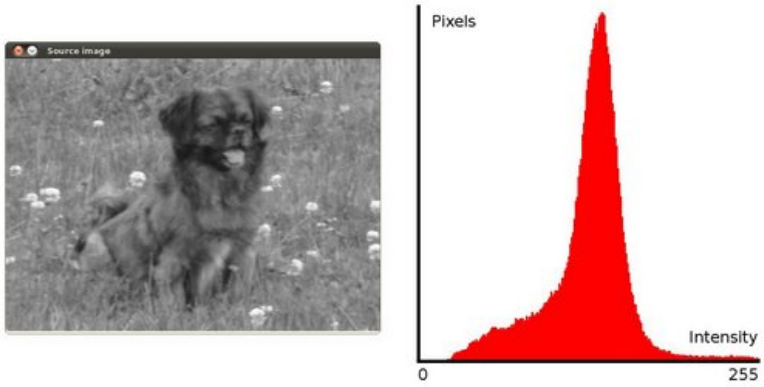
Parametry zkreslení jsou stejné bez ohledu na použité rozlišení kamery.

K aplikování konfigurace na kameru se používá funkce *calibrateCamera* z OpenCV. [14]

3.4.8 Automatické vyvážení obrazu

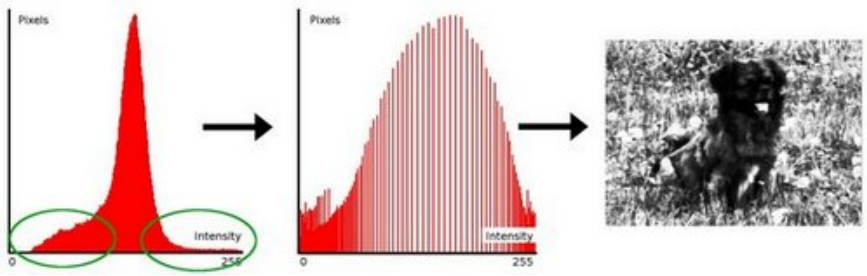
Pokud se pro zachycení obrazu použije varianta webové kamery, je možné využít implementovaný algoritmus pro automatické vyvážení obrazu.

Automatické vyvážení obrazu je založeno na equalizaci histogramu. Histogram je grafické znázornění rozložení intenzity osvětlení obrazu, které kvantifikuje počet pixelů pro jednotlivé hodnoty intenzity. Equalizace histogramu je metoda, která zlepšuje kontrast v obraze pomocí rozšíření rozsahu intenzity.



Obrázek 3.14: Histogram obrázku [17]

Na výše uvedeném obrázku je vidět, že pixely se shlukují zhruba kolem středu dostupného rozsahu intenzity. Equalizace histogramu roztáhne tento rozsah, jak je vidět na obrázku níže.
Zelené elipsy (vlevo) označují oblast s nízkým počtem pixelů. Equalizací došlo k roztažení rozsahu intenzity (uprostřed). Výsledný obrázek je zobrazen vpravo. [17]

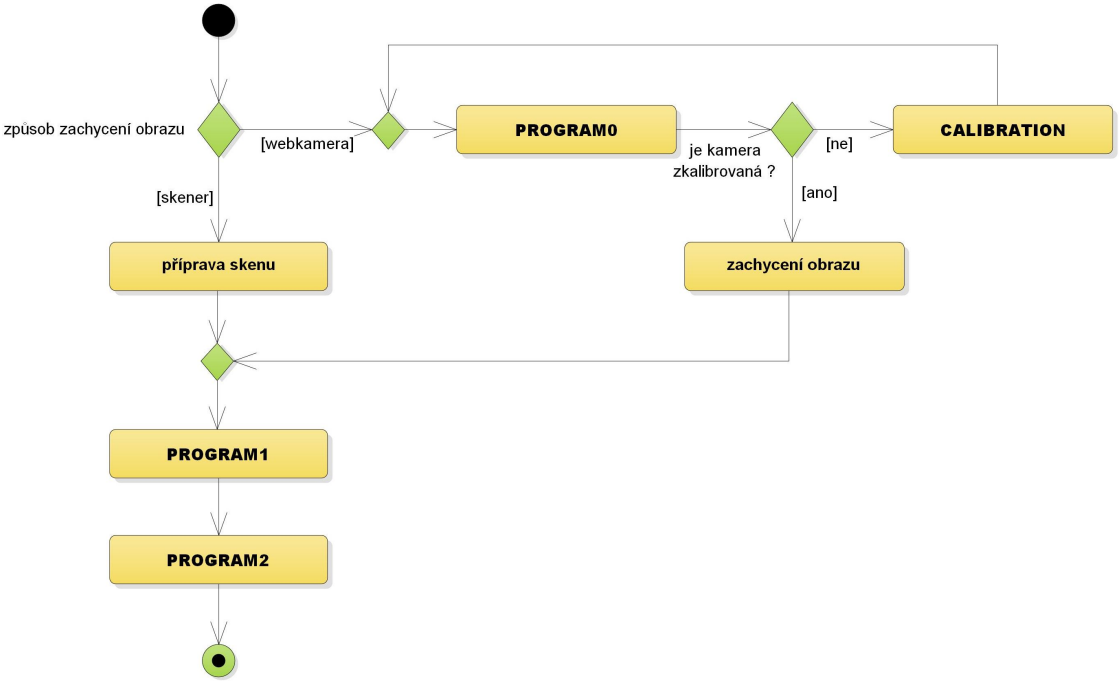


Obrázek 3.15: Equalizace histogramu [17]

3.5 Popis aplikace

Aplikace je napsána v jazyce *C++* s použitím knihovny OpenCV *(Open Source Computer Vision Library - open source knihovna pro počítačové vidění)* [20]
Při implementaci a testování byla použita verze OpenCV 3.2 + balíček opencv_contrib verze 3.2 (balíček extra modulů, dříve součástí distribuce OpenCV, od verze 3.0 mimo) [21]. Z tohoto balíčku byla použita knihovna `xfeatures2d`, která podporuje algoritmus SURF (3.4.5).

Aplikace je rozdělena na čtyři samostatně spustitelné programy (*CALIBRATION, PROGRAM0, PROGRAM1, PROGRAM2*). Workflow je znázorněno na diagramu 3.16.



Obrázek 3.16: Workflow aplikace

V následujících podkapitolách se nachází popis všech programů, u kterých je popsán postup a jsou uvedeny konkrétní algoritmy, které byly použity při jejich implementaci. Dále je také specifikován vstup a výstup programů. Jsou zde také uvedeny příkazy ke spuštění a příklady spuštění.

Struktura celé aplikace je znázorněna v příloze A.

Postup instalace OpenCV a doplňkového balíčku opencv_contrib se nachází v příloze B.

Uživatelská příručka k celé aplikaci je v příloze C.

3.5.1 CALIBRATION - Kalibrace kamery

Popis programu

Program provádí kalibraci webové kamery pomocí kalibračního obrazce. Ten se nachází v přiloženém souboru *pattern.zip* (viz příloha H). Předpokládá se umístění kamery na stativu/stojanu, ideálně v úhlu 90° ke snímanému formuláři. Dále je potřeba zaručit co nejrovnoměrnější osvětlení snímaného formuláře.

Postup programu

V následujících bodech je popsán postup programu, barevně jsou rozlišeny akce programu (●) a akce uživatele (●).

- ● načtení a zpracování konfiguračního souboru *config.ini* (popis v příloze D)
- ● pokus o připojení k webové kameře
  - při úspěchu pokračování v běhu programu
  - při neúspěchu ukončení a zobrazení chybové hlášky
- ● spuštění procesu kalibrace kamery pomocí kalibračního obrazce šachovnice
- ● zobrazení živého náhledu a počtu zachycených snímků (ukázka na obrázku 3.17)
- ● pokud program odhalí kalibrační obrazec v obraze, může uživatel pomocí klávesy *MEZERNÍK* zachytit snímek
  Je potřeba zachytit různé polohy kalibračního obrazce po celé ploše obrazu a s různým natočením, aby došlo ke správné kalibraci!
- ● dokud se počet zachycených snímků nerovná počtu definovanému v konfiguračním souboru (parametr *FRAMES* - defaultně 10), musí uživatel pokračovat v pořizování snímků
- ● dosáhne-li počet zachycených snímků potřebné hranice, dojde ke spuštění algoritmu kalibrace kamery (více o algoritmu v kapitole 3.4.7)
- ● po proběhnutí kalibrace se uloží konfigurační soubor kalibrace kamery (*camera_config.ini*, ukázka souboru v příloze F) a zobrazí se náhled s konfigurovanou (zkalibrovanou) kamerou (ukázka na obrázku 3.18)

Argumenty spuštění

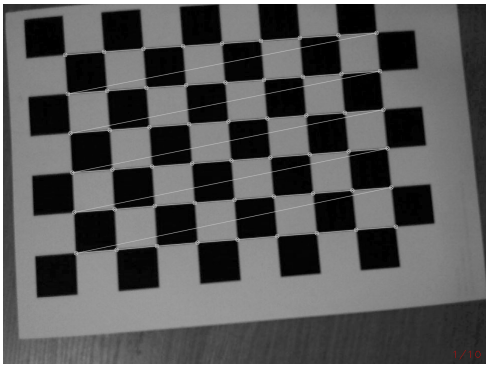
Příkaz. `CALIBRATION [path1] [int1]`
`[path1]` relativní cesta do zpracovávané složky
`[int1]` číslo kamery (připojené kamery jsou číslovány od 0)

Příklad spuštění. `CALIBRATION "../data/20170501" 0`
pro zpracování obrazu z kamery 0 a uložení souboru kalibrace do složky `../data/20170501`

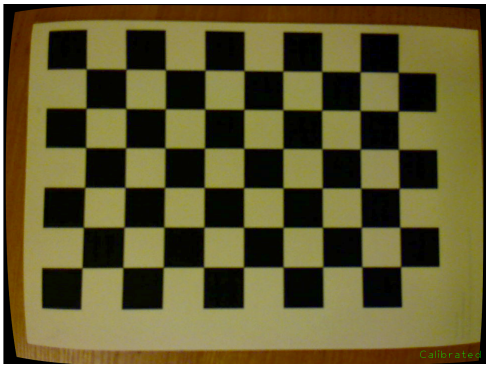
Vstup a výstup

Vstup. obraz z webové kamery

Výstup. konfigurační soubor kalibrace kamery *camera_config.ini*
Výstupní soubor se uloží do složky `[path1]`.
Ukázka souboru se nachází v příloze F.



Obrázek 3.17: Program *CALIBRATION* s nalezeným kalibračním obrazcem



Obrázek 3.18: Program *CALIBRATION* s náhledem po úspěšné kalibraci

3.5.2 PROGRAM0 - Zachycení pomocí webové kamery

Popis programu

Program provádí zachycení obrazu pomocí webové kamery. Webová kamera musí být ovšem nejdříve zkalibrovaná. Stejně jako u programu *CALIBRATION*, i zde se předpokládá umístění kamery na stativu/stojanu, ideálně v úhlu 90° ke snímanému formuláři. Dále je potřeba zaručit co nejrovnoměrnější osvětlení snímaného formuláře.

Postup programu

V následujících bodech je popsán postup programu, barevně jsou rozlišeny akce programu (●) a akce uživatele (●).

- ● načtení a zpracování konfiguračního souboru *config.ini* (popis v příloze D)
- ● pokus o načtení konfiguračního souboru kalibrace kamery *camera_config.ini* (ukázka v příloze F)
  - při úspěchu pokračování v běhu programu
  - při neúspěchu ukončení a zobrazení chybové hlášky
- ● pokus o připojení k webové kameře
  - při úspěchu pokračování v běhu programu
  - při neúspěchu ukončení a zobrazení chybové hlášky
- ● konfigurace kamery z načteného souboru kalibrace
- ● uživatel vidí živý náhled
- ● uživatel může pomocí posuvníků upravit parametry jasu a kontrastu obrazu
- ● pomocí klávesy *A* může uživatel zapnout/vypnout automatické vyvážení obrazu (více o algoritmu v kapitole 3.4.8)
  V tu chvíli jsou ignorovány hodnoty na posuvnících jasu a kontrastu.
- ● pomocí klávesy *MEZERNÍK* uživatel zachytí obraz, který se uloží

Argumenty spuštění

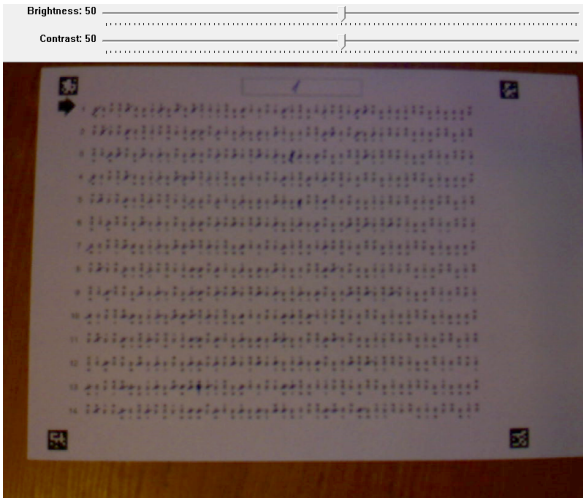
Příkaz. `PROGRAM0 [path1] [int1]`
`[path1]` relativní cesta do zpracovávané složky
`[int1]` číslo kamery (připojené kamery jsou číslovány od 0)

Příklad spuštění. `PROGRAM0 "../data/20170501" 0`
pro zpracování obrazu z kamery 0 a ukládání do složky `../data/20170501/SOURCES`

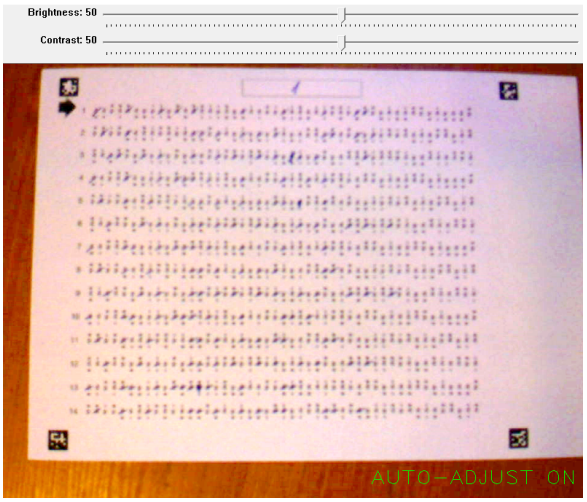
Vstup a výstup

Vstup. obraz z webové kamery

Výstup. zachycený obraz z webové kamery
Výstupní soubor se uloží do složky `[path1]/SOURCES`.
Název souboru bude ve tvaru *PREFIX**.jpg, kde *PREFIX* označuje parametr prefixu z konfiguračního souboru *config.ini* a * označuje nejbližší volné celé číslo pro název souboru (číslováno od 0).



Obrázek 3.19: Grafické rozhraní *PROGRAMu0* s obrazem z webové kamery



Obrázek 3.20: Grafické rozhraní *PROGRAMu0* s obrazem z webové kamery s aktivovaným automatickým vyvážením obrazu (více o algoritmu v 3.4.8)

3.5.3 PROGRAM1 - Transformace zaznamenaného obrazu do referenčního obrazu

Popis programu

Program provádí zaměření a transformaci zaznamenaného obrazu do referenčního obrazu. K tomu využívá referenční obraz uložený v samostatném souboru a algoritmy popsané v kapitole *Návrh a implementace*, podkapitola *Algoritmy* - 3.4.

Postup programu

V následujících bodech je popsán postup programu. Program pracuje bez zásahu uživatele.

- načtení a zpracování konfiguračního souboru *config.ini*
  (popis v příloze D)
- načtení referenčního obrazu *reference.tif* a převedení na obraz v odstínech šedi - *GRAYSCALE*
  Převedení na *GRAYSCALE* se provede pomocí OpenCV funkce *cvtColor*.
- načtení obrazu pro zpracování (z příkladu *1.jpg*) a převedení na *GRAYSCALE*
- změna velikosti načtených obrazů na požadované rozměry pro zpracování
- nalezení významných bodů a výpočet deskriptorů pomocí algoritmu SURF pro oba obrazy
  (více o algoritmu v kapitole 3.4.5)
- párování shod deskriptorů pomocí algoritmu FLANN matcher
  (více o algoritmu v kapitole 3.4.5)
- lokalizace bodů referenčního obrazu v obrazu ke zpracování na základě shod z předchozího kroku
- nalezení transformační matice pomocí funkce OpenCV *findHomography*
  (více o algoritmu v kapitole 3.4.6)
- použití funkce OpenCV *warpPerspective* na původní barevný obraz
  (více o algoritmu v kapitole 3.4.6)
- uložení zpracovaného obrazu ve formátu *JPG*

Argumenty spuštění

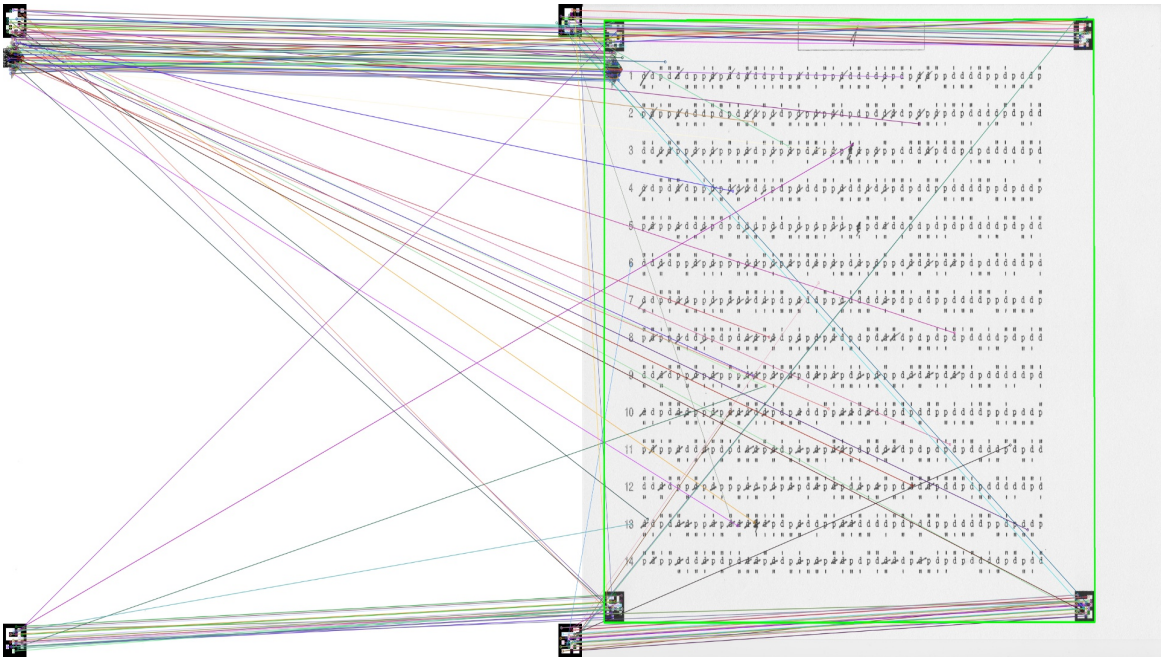
Příkaz. `PROGRAM1 [path1] [string1]`
`[path1]` relativní cesta do zpracovávané složky
`[string1]` název zpracovávaného souboru

Příklad spuštění. `PROGRAM1 "../data/20170501" 1.jpg`
pro zpracování souboru `1.jpg` ze složky `../data/20170501/SOURCES`

Vstup a výstup

Vstup. zachycený obraz ze skeneru nebo webkamery
Obraz se musí nacházet ve složce `[path1]`*/SOURCES*
Doporučené formáty jsou *JPG, BMP, PNG.*

Výstup. zaměřený a transformovaný obraz do referenčního obrazu
Výstupní soubor (s názvem `[string1]`) se uloží do složky `[path1]`*/PREPROCESSED.*



Obrázek 3.21: Ukázka zaměření referenčního obrazu v zaznamenaném obrazu

3.5.4 PROGRAM2 - Detekce odpovědí a vyhodnocení testu

Popis programu

Program provádí detekci odpovědí a vyhodnocení testu. K tomu využívá masku správných odpovědí uloženou v samostatném souboru a algoritmy popsané v kapitole *Návrh a implementace*, podkapitola *Algoritmy* - 3.4. Sejmutí testu je postaveno na detekci barvy psací potřeby, kterou je prováděno zaškrtávání (implementována je varianta pro modrou a červenou barvu).

Postup programu

V následujících bodech je popsán postup programu, barevně jsou rozlišeny akce programu (●) a akce uživatele (●).

Postup programu se liší podle zvoleného režimu.
V manuálním režimu má uživatel možnost upravit zpracovávaný obraz a parametry pro sejmutí testu pomocí posuvníků. Toto nastavení poté může uložit a použít pro automatický režim, kde už uživatel do běhu programu nezasahuje.

Manuální režim

- načtení a zpracování konfiguračního souboru *config.ini* (popis v příloze D)
- pokus o načtení textu poznámky ze souboru *note.txt*
  - text poznámky se objeví v hlavičce souboru s výsledky
- načtení obrazu pro zpracování (z příkladu *1.jpg*)
- změna velikosti obrazu na požadované rozměry pro zpracování
- načtení vyhodnocovací masky ze souboru (*mask.txt*)
- převod BGR na HSV (více o algoritmu v kapitole 3.4.2)
- pokus o načtení nastavení z *program2_options.ini* (ukázka v příloze E)
  - při úspěchu nastavení posuvníků podle souboru s nastavením
  - při neúspěchu nastavení posuvníků na defaultní hodnoty
- uživatel vidí náhledy (aktualizují se při změně parametrů):
  - detekce barvy v obraze (více o algoritmu v kapitole 3.4.1), náhled na obrázku 3.22

    - sejmutí formuláře
      (více o algoritmu v kapitole 3.4.3), náhled na obrázku 3.23

- uživatel může pomocí posuvníků upravit parametry:
    - jasu a kontrastu obrazu
    - minimální a maximální citlivosti detekce
      (maximální citlivost k určení prahu pro opravená zaškrtnutí)
- pomocí klávesy *S* může uživatel uložit aktuální nastavení
  (*program2_options.ini*)
- pomocí klávesy *MEZERNÍK* uživatel sejme a vyhodnotí test
  (více o algoritmu v kapitole 3.4.4).

Automatický režim

- načtení a zpracování konfiguračního souboru *config.ini*
  (popis v příloze D)
- pokus o načtení textu poznámky ze souboru *note.txt*
    - text poznámky se objeví v hlavičce souboru s výsledky
- načtení obrazu pro zpracování (z příkladu *1.jpg*)
- změna velikosti obrazu na požadované rozměry pro zpracování
- načtení vyhodnocovací masky ze souboru (*mask.txt*)
- převod BGR na HSV
  (více o algoritmu v kapitole 3.4.2)
- pokus o načtení nastavení z *program2_options.ini*
  (ukázka v příloze E)
    - při úspěchu pokračování v běhu programu
    - při neúspěchu přepnutí do **Manuálního režimu**
- úprava obrazu a nastavení citlivosti podle parametrů v nastavení
  (*program2_option.ini*)
- detekce barvy v obraze
  (více o algoritmu v kapitole 3.4.1)
- algoritmus pro sejmutí formuláře
  (více o algoritmu v kapitole 3.4.3)
- algoritmus pro vyhodnocení výsledků
  (více o algoritmu v kapitole 3.4.4)

Argumenty spuštění

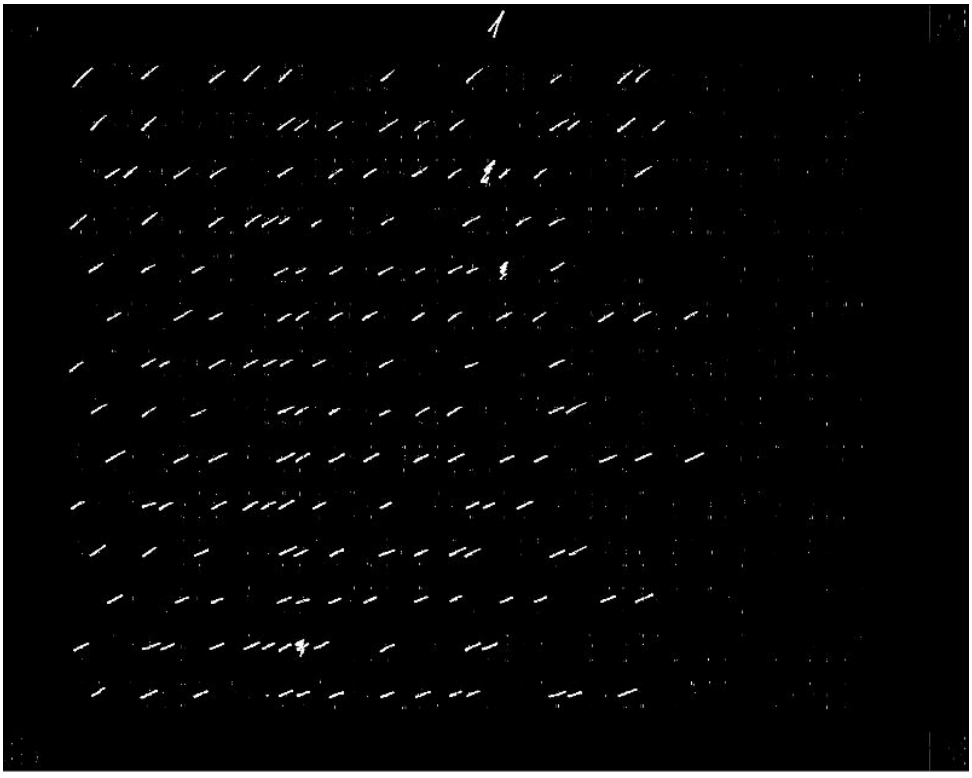
Příkaz. `PROGRAM2 [path1] [string1] [char1] [char2]`
`[path1]` relativní cesta do zpracovávané složky
`[string1]` název zpracovávaného souboru
`[char1]` volby: `R/B` - červená/modrá barva (jakou barvou je formulář vyplněný)
`[char2]` volby: `A/M` - automatický/manuální režim

Příklad spuštění. `PROGRAM2 "../data/20170501" 1.jpg B M`
pro manuální zpracování souboru `1.jpg` ze složky `../data/20170501/PREPROCESSED`, formulář je vyplněn modře

Vstup a výstup

Vstup. zpracovaný obraz (zaměřený a transformovaný) z *PROGRAMu1*
Obraz se musí nacházet ve složce `[path1]`*/PREPROCESSED*

Výstup. soubor s výsledky testu ve formátu *CSV*
Výstupní soubor s výsledky uloží do složky `[path1]`*/RESULTS*.



Obrázek 3.22: Náhled grafického rozhraní *PROGRAMu2* s detekcí barvy v obraze v manuálním režimu

Obrázek 3.23: Náhled grafického rozhraní *PROGRAMu2* s posuvníky v manuálním režimu (červeně orámovaná pole, která byla vyhodnocena jako zaškrtnutá)

Kapitola 4

Testování

V následujících kapitolách následuje testování aplikace a závěrečná analýza.

Testy byly provedeny s 20 vzorky vyplněných d2 formulářů. Formuláře byly vyplněny popisovačem *Centropen LINER 4621 F* o šířce stopy 0,3mm. Polovina vzorků byla vyplněna modrou a polovina červenou barvou.

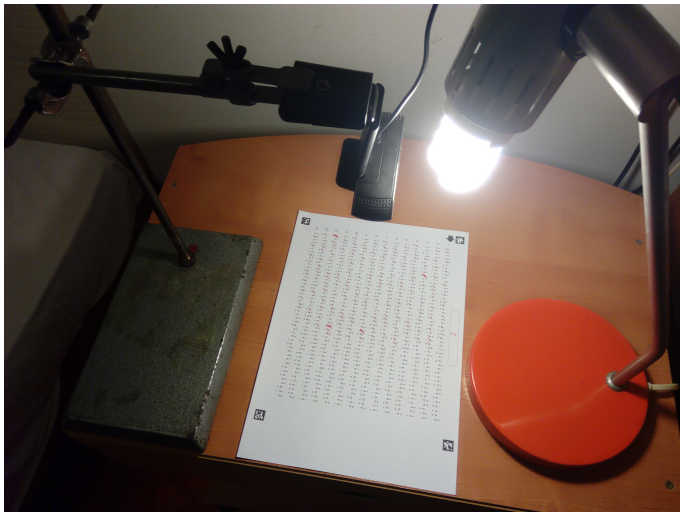
Testovaly se obě varianty aplikace - zachycení pomocí skeneru i pomocí webové kamery.
Pro skenování byl použit skener *Canon MP550* s rozlišením 7016x4960 pixelů. Pro zachycení pomocí webové kamery byla pro testování použita webová kamera *Logitech HD PRO WEBCAM C920* s rozlišením 2304x1536 pixelů.

Testovací data jsou uložena v souboru *tests.zip* (viz příloha H). Struktura je následující:

```
tests/
├── TEST_scanner_blue ... skener, modrý popisovač
│   ├── PREPROCESSED ... zpracované obrazy pomocí PROGRAMu1
│   ├── RESULTS ... výsledky zpracované pomocí PROGRAMu2
│   │   ├── auto ... výsledky testů automatického režimu
│   │   └── manual ... výsledky testů manuálního režimu
│   ├── SOURCES ... skeny
│   ├── note.txt ... soubor s textem poznámky
│   └── program2_options.ini ... soubor s nastavením pro PROGRAM2
├── TEST_scanner_red ... skener, červený popisovač
│   └── podobné jako u TEST_scanner_blue
├── TEST_webcam_blue ... webkamera, modrý popisovač
│   ├── podobné jako u TEST_scanner_blue (v SOURCES jsou zachycené
│   │   obrazy z webkamery)
│   └── + camera_config.ini ... konfigurační soubor kalibrace kamery
└── TEST_webcam_red ... webkamera, červený popisovač
    └── podobné jako u TEST_webcam_blue
```

4.1 PROGRAM0 a CALIBRATION

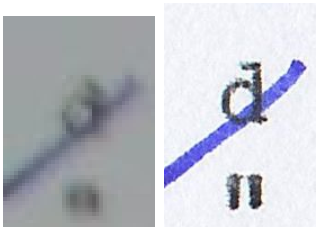
Testování programů *PROGRAM0* a *CALIBRATION* probíhalo s webovou kamerou uchycenou na chemickém stojanu. Jako zdroj světla, které bylo nastaveno tak, aby osvětlovalo formulář co nejrovnoměrněji, sloužila stolní lampička s *13W* úspornou žárovkou (viz obrázek 4.1).



Obrázek 4.1: Testování programů *CALIBRATION* a *PROGRAM0*

Testování kalibrace kamery pomocí kalibračního obrazu šachovnice i zachycení obrazu (včetně možnosti úpravy obrazu před zachycením) proběhlo u všech vzorků úspěšně.

Problémem bylo špatné zaostření kamery. Ve většině případů tedy vznikly neostré snímky. Dalším problémem je relativně nízké rozlišení zachycených snímků, osvětlení obrazu a podání barev. Na obrázku 4.2 je porovnání výřezů z obrazů pořízených webovou kamerou a skenerem.



Obrázek 4.2: Porovnání výřezů z obrazů pořízených webovou kamerou (vlevo) a skenerem (vpravo)

4.2 PROGRAM1

Testování *PROGRAMu1* probíhalo se snímky získanými pomocí skeneru a se snímky získanými pomocí webové kamery v předchozí fázi testování. Výsledky jsou zachyceny v následující tabulce. U každého formuláře je uvedeno, jak test skončil:

***M** - formulář vyplněný modrým popisovačem
***Č** - formulář vyplněný červeným popisovačem
OK - test skončil úspěchem
NOK - test skončil neúspěchem (špatné zaměření)
ERR! - při testu se objevila chyba

ID	SKENER	WEBKAMERA
1M	**OK**	**OK**
2M	**OK**	**OK**
3M	**OK**	**OK**
4M	**OK**	**OK**
5M	**OK**	**OK**
6M	**OK**	**OK**
7M	**OK**	**OK**
8M	**OK**	**OK**
9M	**OK**	**OK**
10M	**OK**	**OK**
1Č	**OK**	**OK**
2Č	**OK**	**OK**
3Č	**OK**	**OK**
4Č	**OK**	**OK**
5Č	**OK**	**OK**
6Č	**OK**	**OK**
7Č	**OK**	**OK**
8Č	**OK**	**OK**
9Č	**OK**	**OK**
10Č	**OK**	**OK**

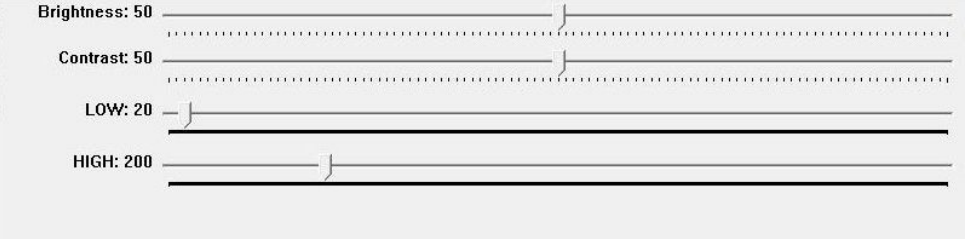
Tabulka 4.1: Výsledky testování *PROGRAMu1*

Z tabulky výše vyplývá, že u všech testovaných vzorků došlo ke správnému zaměření a transformaci obrazu do referenčního obrazu.

4.3 PROGRAM2

Vstupem pro testování *PROGRAMu2* byla výstupní data z předchozí fáze testování (zaměřený a transformovaný obraz).
V rámci testování *PROGRAMu2* byly otestovány obě varianty programu, manuální i automatická.

Parametry pro testy manuálního režimu byly nastavovány v grafickém prostředí programu tak, aby došlo k co nejlepšímu vyhodnocení.



Obrázek 4.3: Posuvníky v grafickém rozhraní *PROGRAMu2*, kterými lze upravit parametry jasu a kontrastu obrazu a parametry citlivosti detekce

Při automatickém režimu byly parametry vyhodnocování nastaveny a uloženy u prvního formuláře a další formuláře byly vyhodnoceny automaticky s uloženými parametry. Parametry pro testy automatického režimu jsou uvedeny v následující tabulce (modrá = modrý popisovač, červená = červený popisovač):

PARAMETR	SKENER		WEBKAMERA	
	modrá	červená	modrá	červená
BRIGHTNESS	50	50	59	50
CONTRAST	50	50	11	34
SENSITIVITY_LOW	75	30	19	45
SENSITIVITY_HIGH	200	200	100	225

Tabulka 4.2: Parametry pro testy automatického režimu *PROGRAMu1*

Výsledky testování jsou zachyceny v následující tabulce. U každého formuláře je uveden počet chybně vyhodnocených políček při testu. Chyby jsou rozděleny do dvou kategorií:
K1 - Nedetekované zaškrtnuté pole a **K2** - Detekované nezaškrtnuté pole.

M - manuální režim, **A** - automatický režim
***M** - formulář vyplněný modrým popisovačem
***Č** - formulář vyplněný červeným popisovačem

ID	SKENER						WEBKAMERA					
	M			A			M			A		
	K1	K2	∑	K1	K2	∑	K1	K2	∑	K1	K2	∑
1M	0	0	**0**	0	0	**0**	21	51	**72**	21	51	**72**
2M	0	0	**0**	0	0	**0**	3	3	**6**	161	0	**161**
3M	0	0	**0**	0	0	**0**	0	2	**2**	187	0	**187**
4M	0	0	**0**	0	0	**0**	0	2	**2**	196	0	**196**
5M	0	0	**0**	0	0	**0**	10	20	**30**	187	0	**187**
6M	0	0	**0**	1	0	**1**	14	37	**51**	134	0	**134**
7M	0	0	**0**	0	0	**0**	7	20	**27**	138	0	**138**
8M	0	0	**0**	1	0	**1**	79	13	**92**	234	0	**234**
9M	0	0	**0**	0	0	**0**	0	0	**0**	158	0	**158**
10M	0	0	**0**	0	0	**0**	3	1	**4**	267	0	**267**
1Č	0	0	**0**	0	0	**0**	0	0	**0**	0	0	**0**
2Č	0	0	**0**	0	0	**0**	0	0	**0**	4	0	**4**
3Č	0	0	**0**	0	0	**0**	0	1	**1**	1	2	**3**
4Č	0	0	**0**	0	0	**0**	1	2	**3**	2	2	**4**
5Č	0	0	**0**	0	0	**0**	0	0	**0**	0	0	**0**
6Č	0	0	**0**	0	0	**0**	17	0	**17**	98	1	**99**
7Č	0	0	**0**	0	0	**0**	6	0	**6**	30	0	**30**
8Č	0	0	**0**	1	0	**1**	129	0	**129**	196	1	**197**
9Č	0	0	**0**	0	0	**0**	53	0	**53**	154	1	**155**
10Č	0	0	**0**	0	0	**0**	47	0	**47**	172	2	**174**

Tabulka 4.3: Výsledky testování *PROGRAMu2*

Výsledky automatického režimu závisí na uloženém nastavení parametrů (nastaveno a uloženo u prvního formuláře). Pokud bychom zvolili jiné parametry, než je uvedeno v tabulce 4.2, dostali bychom odlišné výsledky.

Například u vyhodnocování modře vyplněných formulářů zachycených webovou kamerou se u automatického režimu projevilo špatné nastavení parametrů u prvního formuláře. U dalších poté nebyla detekována žádná zaškrtnutá pole.

Z tabulky 4.3 vyplývá, že nejlépe dopadla varianta zachycení obrazu pomocí skeneru a vyhodnocení pomocí manuálního režimu *PROGRAMu2* bez ohledu na barvu, jakou byl formulář vyplněn. U automatického režimu došlo k chybám v počtu jednotek.
Varianta s webovou kamerou nedosahuje vysoké úspěšnosti v manuálním ani automatickém režimu *PROGRAMu2*. Mírně lepší úspěšnost v tomto případě byla u formulářů vyplněných červenou barvou.

4.4 Závěr testování

Varianta zachycení obrazu pomocí skeneru dopadla v testech výrazně lépe než varianta zachycení pomocí webové kamery (viz tabulka 4.3).
Rozlišení obrazu ze skeneru je vyšší, obraz je zaostřený a má rovnoměrné osvětlení, což vede k lepším výsledkům. Navíc není potřeba provádět kalibraci obrazu, jelikož u skeneru dochází k minimálnímu zkreslení.
Webová kamera použitá při testování nedosahovala rozlišení jako obraz ze skeneru. Pokud bychom chtěli docílit rozlišení alespoň *4K*, byly by náklady na pořízení kamery více než 6000,-. Ale ani při použití webové kamery se stejným rozlišením jako skener nebudeme dosahovat takových výsledků, jako s obrazem ze skeneru.

U automatického režimu *PROGRAMu2* je potřeba stále manuální kontrola výsledků, protože vyplnění formuláře (síla, směr a další vlastnosti zaškrtnutí) se liší u každého člověka a nelze nastavit univerzální parametry, které by zaručily bezchybné vyhodnocení testu.

Na základě výsledků testování lze doporučit:

- **používat variantu zachycení obrazu pomocí skeneru**
- **u *PROGRAMu2* používat manuální režim nebo automatický režim s manuální kontrolou výsledků**
- **používat při vyplňování formulářů psací potřeby použité při testování - *Centropen LINER 4621 F* (nebo podobný typ) o šířce stopy 0,3mm červené nebo modré barvy**

Kapitola 5

Závěr

Cílem této práce bylo vytvořit aplikaci pro automatické vyhodnocování d2 testu, která umožní vyhodnocení vyplněného formuláře pomocí metod počítačového vidění.
Byly popsány algoritmy zpracování obrazu a vyhodnocování zachyceného obrazu, které poté byly implementovány. Řešení umožňuje zpracování obrazu ze skeneru i webové kamery. Byla využita volně dostupná knihovna OpenCV, která zahrnuje rozsáhlou sadu algoritmů pro počítačové vidění.

Aplikace byla poté úspěšně otestována. Nejspolehlivější variantou se ukázalo zachycení obrazu skenerem a jeho následné zpracování. Nepříliš úspěšná pak byla varianta zachycení obrazu pomocí webové kamery. Ta, ve srovnání s variantou se skenerem, podávala špatné výsledky.

Na základě testování a odezvy od koncových uživatelů bude probíhat vývoj i po odevzdání bakalářské práce. Do budoucna se počítá se zdokonalením algoritmů pro vyhodnocování a v plánu je implementace skriptů pro snadnější ovládání aplikace koncovým uživatelem. Dalším plánovaným rozšířením je vyhotovení skriptů pro dávkové zpracování dat.

Literatura

[1] BAY H., TUYTELAARS T., GOOL L.V.. *Surf: Speeded Up Robust Features*. ETH Zurich. 2006

[2] CACEK, PAVEL. *Tvorba panoramatických fotografií*. Brno, 2015. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií, Ústav počítačové grafiky a multimédií.

[3] CAGAŠ, PAVEL. *Extrakce lokálních obrazových deskriptorů na GPU*. Brno, 2014. Bakalářská práce. Masarykova univerzita, Fakulta informatiky.

[4] DANNHOFEROVÁ, Ing. Mgr. Jana. *Vybrané kapitoly z PGR: Barevné prostory*. Mendelova univerzita v Brně: Elektronické studijní materiály [online]. Brno, 2006 [cit. 2017-01-17]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/index.pl?cast=772

[5] GABRHEL, Vít. *Test pozornosti d2: Recenze metody. TESTFÓRUM* [online]., 2014, 3(4), - [cit. 2017-01-14]. DOI: 10.5817/TF2014-4-26. ISSN 18059147. Dostupné z: http://testforum.cz/domains/testforum.cz/index.php/testforum/article/view/26

[6] KÖHLER J., PAGANI A., STRICKER D. *Detection and Identification Techniques for Markers Used in Computer Vision*. Department of Augmented Vision. German Research Center for Artificial Intelligence GmbH. Trippstadter Str. 122, 67663 Kaiserslautern, Germany

[7] KULA, JIŘÍ. *Kalibrace multi-kamerového systému*. Praha, 2012. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra počítačů.

[8] ŠVÁB, JAN. *Akcelerace zpracování obrazu hradlovým polem*. Praha, 2009. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra kybernetiky.

[9] TRÁVNÍČEK, VOJTĚCH. *Porovnání významných bodů pro detekci objektů v obraze*. Brno, 2013. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínského inženýrství.

[10] Integrální obraz. In: Wikipedia: the free encyclopedia [online]., San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-01-18]. Dostupné z: https://cs.wikipedia.org/wiki/Integr%C3%A1ln%C3%AD_obraz

[11] OpenCV library: About. *OpenCV library* [online]., 2016 [cit. 2017-05-01]. Dostupné z: Dostupné z: http://opencv.org/about.html

[12] SFÉRICKÁ VADA. *Foto Roman* [online]., [cit. 2017-01-19]. Dostupné z: http://fotoroman.cz/glossary/2_sfera.htm

[13] SURF. In: Wikipedia: the free encyclopedia [online]., San Francisco (CA): Wikimedia Foundation,[cit. 2017-01-18]. Dostupné z: https://cs.wikipedia.org/wiki/SURF

Dokumentace OpenCV

[14] Camera calibration With OpenCV. *Camera calibration With OpenCV* [online]., [cit. 2017-01-19]. Dostupné z: http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

[15] findHomography. *OpenCV: Camera Calibration and 3D Reconstruction* [online]., 2017 [cit. 2017-01-19]. Dostupné z: http://docs.opencv.org/trunk/d9/d0c/group___calib3d.html

[16] FLANN matcher. *cv::FlannBasedMatcher Class Reference* [online]., 2017 [cit. 2017-01-18]. Dostupné z: http://docs.opencv.org/trunk/dc/de2/classcv_1_1FlannBasedMatcher.html

[17] Histogram Equalization. *Histogram Equalization* [online]., [cit. 2017-05-01]. Dostupné z: http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html

[18] Histograms. *Histograms* [online]., [cit. 2017-05-01]. Dostupné z: http://docs.opencv.org/2.4/modules/imgproc/doc/histograms.html

[19] inRange. *OpenCV: Operation on arrays* [online]., 2016 [cit. 2017-01-19]. Dostupné z: http://docs.opencv.org/3.2.0/d2/de8/group___core___array.html

[20] OpenCV. *OpenCV* [online]., Dostupné z: http://opencv.org/

[21] Opencv_contrib [online]., Dostupné z: https://github.com/opencv/opencv_contrib

[22] SURF. *cv::xfeatures2d::SURF Class Reference* [online]., 2015 [cit. 2017-01-18]. Dostupné z: http://docs.opencv.org/3.1.0/d5/df7/classcv_1_1xfeatures2d_1_1SURF.html

[23] warpPerspective. *OpenCV: Geometric Image Transformations* [online]., 2017 [cit. 2017-01-19]. Dostupné z: http://docs.opencv.org/trunk/da/d54/group___imgproc___transform.html

Obrázky

[24] CV markers. [online]., [cit. 2017-05-15]. Dostupné z: http://doi.ieeecomputersociety.org/cms/Computer.org/dl/trans/tp/2010/07/figures/ttp20100713173.gif

[25] Feature Detection and Matching. *Feature Detection and Matching* [online]., [cit. 2017-01-19]. Dostupné z: https://courses.cs.washington.edu/courses/cse576/13sp/projects/project1/artifacts/woodrc/index.htm

[26] Feature Matching with FLANN. *Feature Matching with FLANN* [online]., [cit. 2017-05-01]. Dostupné z: http://docs.opencv.org/2.4/doc/tutorials/features2d/feature_flann_matcher/feature_flann_matcher.html

[27] File:AdditiveColor.svg. In: Wikipedia: the free encyclopedia [online]., San Francisco (CA): Wikimedia Foundation, 2006 [cit. 2017-01-19]. Dostupné z: https://en.wikipedia.org/wiki/File:AdditiveColor.svg

[28] File:HSV cone.png. In: Wikipedia: the free encyclopedia [online]., San Francisco (CA): Wikimedia Foundation, [cit. 2017-01-19]. Dostupné z: https://commons.wikimedia.org/wiki/File:HSV_cone.png

[29] File:Integralni obraz.png. In: Wikipedia: the free encyclopedia [online]., San Francisco (CA): Wikimedia Foundation, 2008 [cit. 2017-01-19]. Dostupné z: https://commons.wikimedia.org/wiki/File:Integralni_obraz.png

[30] OpenCV calibration pattern. *Camera calibration With OpenCV.* [online]., [cit. 2017-05-15] Dostupné z: http://docs.opencv.org/2.4/_downloads/pattern.png

[31] OpenCV: Detection of Diamond Markers. OpenCV 3.1.0 Open Source Computer Vision [online]., 2015 [cit. 2017-01-19]. Dostupné z: http://docs.opencv.org/3.1.0/d5/d07/tutorial_charuco_diamond_detection.html

Příloha A

Struktura aplikace

```
/
├── app ... složka s jednotlivými programy
│   ├── CALIBRATION
│   ├── PROGRAM0
│   ├── PROGRAM1
│   ├── PROGRAM2
│   └── LICENSE.txt ... licenční ujednání (viz kapitola 3.2)
└── data
    ├── config
    │   ├── config.ini ... konfigurační soubor (viz příloha D)
    │   ├── mask.txt ... maska se správnými odpověďmi pro zpracování
    │   │                PROGRAMem2
    │   └── reference.tif ... referenční obraz pro zpracování PROGRAMem1
    └── YYYYMMDD ... složka s daty (data ke zpracování a výsledky,
        │            doporučený formát YYYYMMDD (rok, měsíc, den))
        ├── SOURCES ... složka s obrazovými daty ze skeneru/webkamery,
        │   │           připravenými pro PROGRAM1
        │   ├── 1.jpg
        │   └── 2.jpg
        ├── PREPROCESSED ... složka s upravenými obrazovými daty,
        │   │                připravenými pro PROGRAM2
        │   ├── 1.jpg
        │   └── 2.jpg
        ├── RESULTS ... složka s uloženými výsledky
        │   ├── 1.csv
        │   └── 2.csv
        ├── camera_config.ini ... konfigurační soubor kalibrace kamery
        ├── note.txt ... soubor s textem poznámky (volitelný)
        └── program2_options.ini ... soubor s nastavením pro PROGRAM2
```

Příloha B

Instalace OpenCV a balíčku opencv_contrib

B.0.1 Linux

Instalace

Tento návod byl převzat a upraven z:
`http://docs.opencv.org/trunk/d7/d9f/tutorial_linux_install.html`
`https://github.com/opencv/opencv_contrib`
v případě problémů navštivte originální návody (EN)

1. balíčky potřebné k instalaci OpenCV nainstalujte následujícími příkazy:
   a. `sudo apt-get install build-essential`
   b. `sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev`
   c. `sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev`
2. vytvořte složku ***opencv-3.2*** a v ní:
   a. rozbalte balíček openCV.zip, extrahovanou složku *opencv-3.2.0* přejmenujte na ***opencv***
   (přiložený soubor - viz příloha H)
   b. rozbalte balíček openCV_contrib.zip, extrahovanou složku *opencv_contrib-3.2.0* přejmenujte na ***opencvC***
   (přiložený soubor - viz příloha H)
3. vstupte do složky ***opencv***, vytvořte složku ***build*** a vstupte do ní:
   `cd opencv`
   `mkdir build`
   `cd build`

4. Spusťte cmake následujícím příkazem:
   `cmake -D OPENCV_EXTRA_MODULES_PATH=../../opencvC/modules ..`

   pokud příkaz nebude fungovat, použijte následující:
   `cmake -DOPENCV_EXTRA_MODULES_PATH=../../opencvC/modules ..`

5. spusťte program *make* (doporučeno je spuštění v 7 vláknech):
   `make -j7`

6. pro instalaci knihoven spusťte následující příkaz:
   `sudo make install`

Kompilace

Tento návod byl převzat a upraven z:
`http://docs.opencv.org/trunk/db/df5/tutorial_linux_gcc_cmake.html`
v případě problémů navštivte originální návody (EN)

1. ve složce se zdrojovými kódy vytvořte soubor *CMakeLists.txt* s následujícím obsahem:
   ```
   cmake_minimum_required(VERSION 2.8)
   project( CALIBRATION )
   find_package( OpenCV REQUIRED )
   include_directories( $OpenCV_INCLUDE_DIRS )
   add_executable( CALIBRATION Calibration.cpp )
   target_link_libraries( CALIBRATION $OpenCV_LIBS )
   ```

2. spusťte *cmake* a program *make* následujícími příkazy:
   ```
   cmake .
   make
   ```
   zkompiluje se spustitelný program

3. **body 1 a 2 opakujte pro všechny programy**:
   CALIBRATION, *PROGRAM0*, *PROGRAM1*, *PROGRAM2*
   vždy nahraďte na 2., 5. a 6. řádku název projektu, název programu a název souboru **.cpp* požadovaným programem

B.0.2 Windows

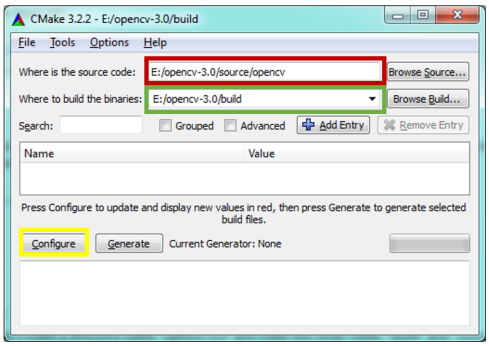
Tento návod byl převzat (včetně obrázků) a upraven z:
`https://putuyuwono.wordpress.com/2015/04/23/building-and-installing-opencv-3-0-on-windows-7-64-bit/`
v případě problémů navštivte návod výše, nebo oficiální návody uvedené níže (EN)
`http://docs.opencv.org/trunk/d3/d52/tutorial_windows_install.html`
`http://docs.opencv.org/trunk/d6/d8a/tutorial_windows_visual_studio_Opencv.html`

je potřeba mít nainstalováno:

- Microsoft Visual Studio
- CMake pro Windows (`http://www.cmake.org/download/`)

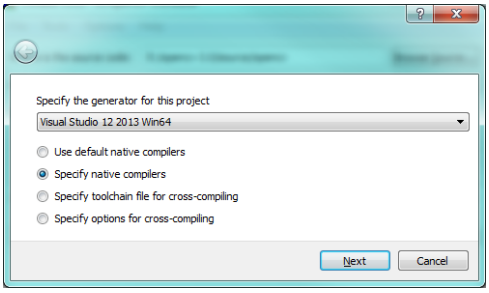
v obrázcích se objevuje složka *opencv-3.0*, namísto ní budeme mít složku *opencv-3.2*

1. vytvořte složku ***opencv-3.2***, ve které vytvořte dvě podsložky:
   a. ***build***
   b. ***source***
2. ve složky ***source***:
   a. rozbalte balíček openCV.zip, extrahovanou složku *opencv-3.2.0* přejmenujte na ***opencv***
   (přiložený soubor - viz příloha H)
   b. rozbalte balíček openCV_contrib.zip, extrahovanou složku *opencv_contrib-3.2.0* přejmenujte na ***opencv_contrib***
   (přiložený soubor - viz příloha H)
3. spusťte *CMake* a specifikujte cesty do složky ***opencv*** (červený rámeček) a ***build*** (zelený rámeček) (obrázek B.1)



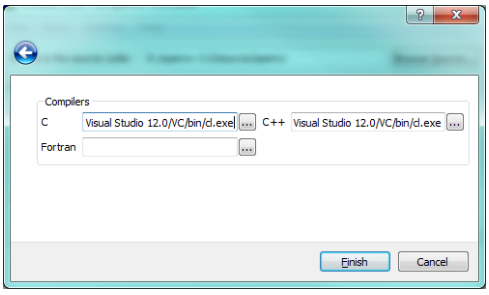
Obrázek B.1: *CMake* nastavení cest zdrojové a cílové složky

4. klikněte na tlačítko *Configure* a specifikujte svoje Microsoft Visual Studio (obrázek B.2)



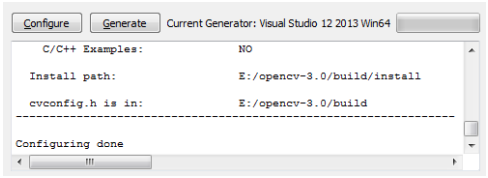
Obrázek B.2: *CMake* specifikace Microsoft Visual Studio

5. klikněte na tlačítko *Next* a specifikujte C a C++ compiler (můžeme využít compiler umístěný v `{Visual Studio}/VC/bin/cl.exe`) (obrázek B.3)



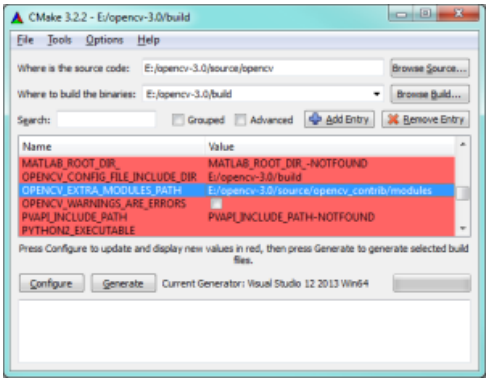
Obrázek B.3: *CMake* specifikace C a C++ compiler

6. klikněte na tlačítko *Finish*, po dokončení práce by se v konzoli měla objevit informace o úspěšné konfiguraci (obrázek B.4)



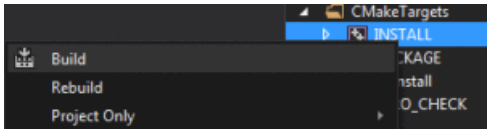
Obrázek B.4: *CMake* konzole s logem s informací o úspěchu konfigurace

7. nyní je potřeba specifikovat cestu k extra modulům `{opencv-3.2}/source/opencv_contrib/modules` (obrázek B.5)



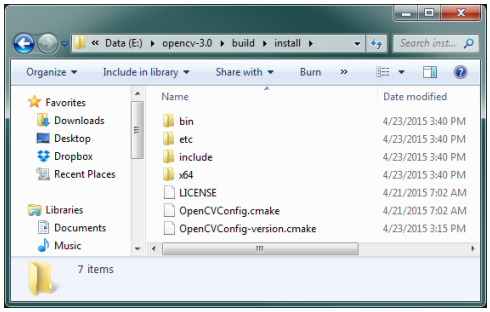
Obrázek B.5: *CMake* specifikace cesty k extra modulům

8. podívejte se do konfiguračního logu v konzoli a najděte položku `OpenCV modules - To be built` a zkontrolujte, zda se v seznamu nachází modul `xfeatures2d`
   • pokud se v seznamu požadovaný modul nenachází, nalezněte ho v nabídce a zaškrtněte checkbox
9. pokud je vše v pořádku, klikněte na tlačítko *Generate*
10. spusťte soubor *OpenCV.sln* ve složce ***build***
11. po načtení projektu najděte projekt *INSTALL* uvnitř složky *CMakeTargets* a zvolte volbu *Build* (pod pravým tlačítkem myši) (obrázek B.6)
    • *Debug/Release* verzi vytvoříme volbou build konfigurace projektu



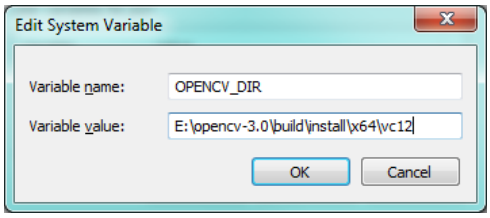
Obrázek B.6: Microsoft Visual Studio, build projektu

12. po úspěšném dokončení procesu se vytvoří nové složky ve složce ***build*** (*install*, *bin*, *lib*) (obrázek B.7 - detail složky *install*)



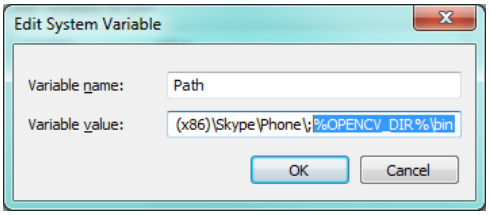
Obrázek B.7: Detail složky *install*

13. otevřete nastavení proměnných prostředí a přidejte novou proměnnou prostředí *OPENCV_DIR* s hodnotou cesty do složky, která se vytvořila ve složce ***build/install/*** viz obrázek B.8 (bude se lišit podle verze Microsoft Visual Studio a systému)



Obrázek B.8: Úprava proměnné prostředí *OPENCV_DIR*

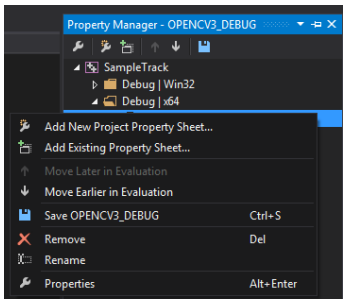
14. do proměnné prostředí *Path* přidejte řetězec *%OPENCV_DIR%\bin* (obrázek B.9)



Obrázek B.9: Úprava proměnné prostředí *Path*

15. vytvořte nový Microsoft Visual Studio C++ projekt

16. otevřete *Property Manager View*, vyberte požadovanou složku a zvolte volbu *Add New Project Property Sheet* (pod pravým tlačítkem myši) (obrázek B.10)
    - *Project Property Sheet* má výhodu v tom, že ke každému novému projektu lze připojit už nastavený *Sheet* a není potřeba znovu absolvovat celé nastavení



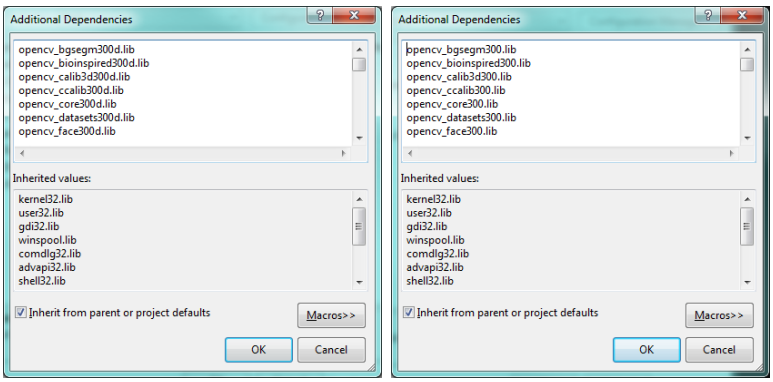
Obrázek B.10: Microsoft Visual Studio, volba *Project Property Sheet*

16. zvolte jméno *Sheetu* a klikněte na tlačítko *Add*

17. pravým tlačítkem myši klikněte na právě vytvořený *Sheet* a zvolte volbu *Properties*

18. nastavte *Additional Include Libraries (Common Properties-C/C++/General)* přidáním záznamu: *$(OPENCV_DIR)\..\..\include*

19. přidejte do *Additional Dependencies (Common Properties-Linker-Input)* všechny soubory ze složky *%(OPENCV_DIR)\lib* (obrázek B.11)
    - Debug mód: pouze soubory končící na *d*
    - Release mód: pouze soubory nekončící na *d*



Obrázek B.11: Microsoft Visual Studio, přidání *Additional Dependencies* vlevo Debug mód, vpravo Release mód

20. nyní už stačí pouze přidat do projektu zdrojový kód a udělat build programu (pro každý program vytvořit samostatný projekt s příslušným názvem *CALIBRATION, PROGRAM0, atd.*)

21. do každého projektu importujte vytvořený *Project Property Sheet* (jako v bodě 16, pouze zvolit možnost *Add Existing Property Sheet*)

Příloha C

Uživatelská příručka

Uživatelská příručka popisuje sled akcí, které je potřeba provést k úspěšnému získání výsledků testu d2 (zachycení, zpracování, vyhodnocení). Předpokládá se standardní struktura aplikace (znázorněna v příloze A).

C.0.1 VARIANTA SKENER

1. vytvořte složku pro zpracování ve struktuře aplikace
   (nová podsložka ve složce *data*, například *20170501*)
2. v nově vytvořené složce vytvořte podsložku *SOURCES* a umístěte do ní naskenované snímky
   (názvy souborů budou použity k pojmenování souborů výsledků a budou doplněny také do hlavičky souboru s výsledky)
3. spusťte *PROGRAM1* s příslušnými argumenty (více v kapitole popisující *PROGRAM1* - 3.5.3)

   - příklad: `PROGRAM1 "../data/20170501" 1.jpg`
     pro zpracování souboru `1.jpg` ze složky `../data/20170501/SOURCES`
   - zpracovaný snímek se uloží do složky `../data/20170501/PREPROCESSED`
4. pokud chcete do souboru výsledku vyplnit do hlavičky poznámku, vytvořte soubor *note.txt* (ve zpracovávané složce) s textem poznámky

Manuální režim

5. spusťte *PROGRAM2* s příslušnými argumenty (více v kapitole popisující *PROGRAM2* - 3.5.4)

   - příklad: `PROGRAM2 "../data/20170501" 1.jpg B M`
     pro manuální zpracování souboru `1.jpg` ze složky `../data/20170501/PREPROCESSED`, formulář je vyplněn modře
6. pomocí posuvníků můžete v grafickém prostředí upravovat:

   a. parametry jasu a kontrastu výsledného obrazu

b. parametry pro minimální a maximální detekci zaškrtnutí (pole s počtem nenulových pixelů ohraničeným těmito dvěma parametry budou vyhodnocena jako zaškrtnutá)

7. klávesou *S* můžete uložit nastavení aktuálních hodnot posuvníků pro další používání programu v této složce (pro zpracování dalších souborů nebo pro automatický režim)

   - nastavení (soubor *program2_options.ini*) se uloží do zpracovávané složky

8. klávesou *MEZERNÍK* dojde k sejmutí testu a výsledky se uloží do podsložky *RESULTS* ve zpracovávané složce

Automatický režim

5. spusťte *PROGRAM2* s příslušnými argumenty (více v kapitole popisující *PROGRAM2* - 3.5.4)
   Je potřeba mít uložené nastavení zmíněné v bodě 7 manuálního režimu.
   Pokud nastavení uložené není (soubor *program2_options.ini*), pokračujte bodem 5 u manuálního režimu.

   - příklad: `PROGRAM2 "../data/20170501" 1.jpg B A`
     pro automatické zpracování souboru `1.jpg` ze složky `../data/20170501/PREPROCESSED`, formulář je vyplněn modře
   - program proběhne automaticky bez zásahu uživatele
   - výsledky se uloží do podsložky *RESULTS* ve zpracovávané složce

C.0.2 VARIANTA WEBKAMERA

1. vytvořte složku pro zpracování ve struktuře aplikace
   (nová podsložka ve složce *data*, například *20170501*)

2. spusťte program *CALIBRATION* s příslušnými argumenty (více v kapitole popisující program *CALIBRATION* - 3.5.1)

   - příklad: `CALIBRATION "../data/20170501" 0`
     pro zpracování obrazu z kamery 0 a uložení souboru kalibrace do složky `../data/20170501`

3. proveďte kalibraci kamery pomocí kalibračního obrazce
   (více v kapitole popisující program *CALIBRATION* - 3.5.1)
   Je potřeba zachytit různé polohy kalibračního obrazce po celé ploše obrazu a s různým natočením, aby došlo ke správné kalibraci!

4. klávesou *MEZERNÍK* zachytíte snímek

   • dosáhne-li počet zachycených snímků požadovaný počet pro kalibraci (počet je zobrazený v okně programu), uloží se kalibrace do zpracovávané složky (soubor *camera_config.ini*)

5. spusťte *PROGRAM0* s příslušnými argumenty (více v kapitole popisující *PROGRAM0* - 3.5.2)

   • příklad: `PROGRAM0 "../data/20170501" 0`
   pro zpracování obrazu z kamery 0 a ukládání do složky
   `../data/20170501/SOURCES`

6. pomocí posuvníků můžete v grafickém prostředí upravovat parametry jasu a kontrastu výsledného obrazu
   klávesou *A* lze zapnout/vypnout automatickou korekci jasu a kontrastu

7. klávesou *MEZERNÍK* zachytíte snímek

   • snímek se uloží do složky *SOURCES* ve zpracovávané složce

8. pokračujte bodem 3 ve **VARIANTĚ SKENER**

Příloha D

Konfigurační soubor *config.ini*

Obsah souboru *config.ini* a překlad komentářů.

Společná část pro všechny programy

```
[PRO VSECHNY PROGRAMY]
;ZACATEK BLOKU, KTERY NEUPRAVOVAT
;rozmery obrazu
IMG_WIDTH=2000
IMG_HEIGHT=1584
;rozmery nahledu
PREVIEW_WIDTH=800
PREVIEW_HEIGHT=634
;KONEC BLOKU, KTERY NEUPRAVOVAT
```

Část pro PROGRAM0

```
[POUZE PROGRAM0]
;prefix pro nazev souboru zachyceneho obrazu
PREFIX=WEBCAM_
```

Část pro CALIBRATION

```
[POUZE CALIBRATION]
;ZACATEK BLOKU, KTERY NEUPRAVOVAT
;specifikace kalibracniho obrazce (sachovnice)
;pocet rohu horizontalne/vertikalne
CORNERS_HORIZONTAL=9
CORNERS_VERTICAL=6
;KONEC BLOKU, KTERY NEUPRAVOVAT
;pocet kalibracnich snimku
FRAMES=10
```

Část pro PROGRAM2

```
[POUZE PROGRAM2]
;ZACATEK BLOKU, KTERY NEUPRAVOVAT
;pocatecni souradnice (levy horni roh prvniho radku)
INIT_X=148
INIT_Y=137
;vzdalenosti pro proces parsovani poli
X_DISTANCE=35
Y_DISTANCE=98
;rozmery poli
FIELD_WIDTH=26
FIELD_HEIGHT=26
;kazdych 11 poli je pridan koeficient (pro spravne parsovani)
COEFFICIENT_X=2
;KONEC BLOKU, KTERY NEUPRAVOVAT
;prahy barevne detekce
HUE_LOW_THRESHOLD_BLUE=110
HUE_LOW_THRESHOLD_RED=170
SATURATION_LOW_THRESHOLD=50
VALUE_LOW_THRESHOLD=50
HUE_HIGH_THRESHOLD_BLUE=130
HUE_HIGH_THRESHOLD_RED=180
SATURATION_HIGH_THRESHOLD=255
VALUE_HIGH_THRESHOLD=255
```

Příloha E

Soubor nastavení pro PROGRAM2 *program2_options.ini*

Ukázka obsahu souboru s nastavením pro *PROGRAM2*.

```
;options for PROGRAM2
BRIGHTNESS=50
CONTRAST=50
SENSITIVITY_LOW=20
SENSITIVITY_HIGH=200
```

BRIGHTNESS parametr jasu

CONTRAST parametr kontrastu

SENSITIVITY_LOW parametr minimální citlivosti detekce

SENSITIVITY_HIGH parametr maximální citlivosti detekce

Příloha F

Konfigurační soubor kalibrace kamery *camera_config.ini*

Ukázka obsahu konfiguračního souboru kalibrace kamery.

```
;camera calibration configuration
DC1=0.120044
DC2=-0.213812
DC3=0.0010563
DC4=0.00254338
DC5=0.0659335
CM11=1685.73
CM12=0
CM13=1142.74
CM21=0
CM22=1695.57
CM23=769.26
CM31=0
CM32=0
CM33=1
```

Jednotlivé řádky představují prvky matic pro kalibraci kamery (viz kapitola 3.4.7).

Matice parametrů zkreslení

$$\begin{pmatrix} DC1 & DC2 & DC3 & DC4 & DC5 \end{pmatrix}$$

Matice kamery

$$\begin{pmatrix} CM11 & CM12 & CM13 \\ CM21 & CM22 & CM23 \\ CM31 & CM32 & CM33 \end{pmatrix}$$

Příloha G

Ukázka výstupního souboru ve formátu CSV

Ukázka výstupního souboru ve formátu CSV.

```
FILE: ../data/20170501/RESULTS/1.csv
NOTE: 01.05.2017 scanner mode
LINE;CORRECT_CROSS;MISTAKE_CROSS;OMIT_CROSS;LAST_CROSS;
1;17;0;4;46;
2;22;0;0;47;
3;16;0;5;45;
4;14;0;7;46;
5;17;0;5;47;
6;15;1;6;46;
7;18;0;3;46;
8;19;0;3;46;
9;19;0;2;47;
10;14;0;7;46;
11;19;0;3;47;
12;20;1;1;47;
13;16;0;5;46;
14;16;1;6;47;
```

FILE název souboru a jeho umístění (pro snadnější identifikaci)

NOTE poznámka (pokud existuje soubor *note.txt* s textem poznámky ve složce, která se zpracovává)

LINE číslo řádky (indexováno od 1)

CORRECT_CROSS počet správně zaškrtnutých polí

MISTAKE_CROSS počet nesprávně zaškrtnutých polí

OMIT_CROSS počet přeskočených polí (měla být zaškrtnuta)

LAST_CROSS index posledního zaškrtnutí (indexováno od 1)

Příloha H

Odevzdané soubory

Seznam souborů, které jsou přiloženy (online):

app.zip balíček se zdrojovými kódy a základní strukturou aplikace

imgs.zip balíček s obrázky prezentující dosažené výsledky

pattern.zip balíček s kalibračním obrazcem [30] ve formátech *PNG*,*PDF*

openCV.zip balíček s knihovnou OpenCV verze 3.2 [20]
(z důvodu omezení velikosti přílohy rozděleno na více částí)

openCV_contrib.zip balíček opencv_contrib verze 3.2 [21]
(balíček extra modulů)
(z důvodu omezení velikosti přílohy rozděleno na více částí)

tests.zip balíček s testovacími daty
(z důvodu omezení velikosti přílohy rozděleno na více částí)