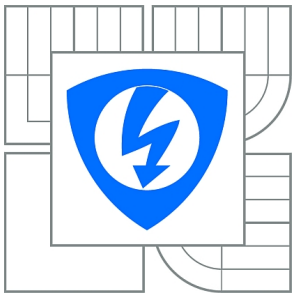
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

AUTOMATICKÁ KONTROLA SPRÁVNOSTI SESTAVENÍ VÝROBKU

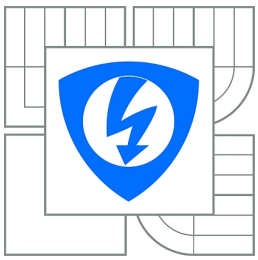
AUTOMATIC VERIFICATION OF PRODUCT ASSEMBLING CORRECTNESS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. PETR DOLEŽAL
AUTHOR

VEDOUCÍ PRÁCE — Ing. KAMIL ŘÍHA, Ph.D.
SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Petr Doležal	***ID:***	84427
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Automatická kontrola správnosti sestavení výrobku

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je návrh aplikace pro kontrolu správnosti sestavení výrobku pomocí metod zpracování digitálních obrazů. Výsledkem by mělo být samostatné pracoviště, do kterého bude vstupovat výrobek spolu s informací, o jakou zákaznickou variantu se jedná. Pracoviště bude obsahovat počítač a jednotku pro pořízení snímků. Snímek výrobku bude vyhodnocen pomocí počítače, výstupem bude informace, zda definovaný výrobek odpovídá požadavkům na jednotlivé atributy dané v tabulce. Výrobek je možné zafixovat, aby byly orientační body na výrobku vždy ve stejné pozici, v rámci návrhu je však předpokládáno ověření možnosti volné pozice analyzovaného výrobku. Pracoviště může být světlotěsné s vlastním světelným zdrojem k dosažení dostatečné spolehlivosti, opět je však předpokládáno ověření možnosti funkce aplikace bez odstínění okolních světelných zdrojů. Spolehlivost detekce všech parametrů je požadována 99% nebo vyšší. Softwarové řešení je možné zvolit, doporučené je vývojové prostředí Microsoft Visual Studio. Je možné využití takových knihoven, které mohou být aplikovány v komerčních podmínkách. Nutná je znalost příslušných licenčních podmínek. Výsledný systém by měl umožňovat případné pozdější úpravy a rozšíření.

DOPORUČENÁ LITERATURA:

[1] GONZALEZ R. C., WOODS R. E.: Digital Image Processing, Prentice Hall, New Jersey, 2002
[2] BRADSKI G., KAEHLER A.: Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, Inc. USA 2008, ISBN: 978-0-596-51613-0
[3] PRATA S.: Mistrovství v C++, Computer Press, Brno 2004, ISBN 80-251-0098-7

Termín zadání:	29.1.2010	***Termín odevzdání:***	26.5.2010

Vedoucí práce: Ing. Kamil Říha, Ph.D.

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

Abstrakt

Tato práce navrhuje softwarový algoritmus pro kontrolu klíčových vlastností výrobku na základě zpracování obrazových dat. Úvodem je popsána motivace, která předcházela jejímu vzniku. Následuje teoretický rozbor použitých pokročilých metod zpracování obrazu – Houghova transformace kružnic, mechanismus semínkového zaplavování. Autor přichází s vlastním mechanismem kompenzace nerovnoměrného osvětlení ve snímané scéně, který je založen na modelování ploch pomocí matematického aparátu Bézierových bikubik. Nechybí popis implementace v jazyce C/C++, metoda kompenzace osvětlení je implementována i prostředí MATLAB. Algoritmus je hodnocen pomocí procentuální úspěšnosti rozpoznání požadovaných vlastností. Důraz autor klade na časovou efektivitu realizace.

Klíčová slova

Zpracování obrazu – OpenCV – semínkové zaplaování – Houghova transformace – Bézierova plocha – detekce – klasifikace

Abstract

This diploma evaluates methods for verification of key characteristics of a product using digital image processing techniques. At first, reasons why this work has been done are described followed by a list of all methods that were used in this diploma such as Hough Circle Transform and Flood Fill (Seed Fill) algorithm. Also, a new approach how to compensate non regularly illuminated scene, which is based on surface modeling with Bézier Surfaces, was developed. Moreover, the algorithm was implemented in the C++ programming language and some of the parts were also simulated using the MATLAB environment. The algorithm was evaluated based on the percentage level of recognition of the required parameters. Efficiency of the implementation is also important for the author.

Keywords

Image processing – OpenCV – Flood Fill – Hough Transform – Bézier Surface – detection – classification

DOLEŽAL, P. *Automatická kontrola správnosti sestavení výrobku.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 54 s. Vedoucí diplomové práce Ing. Kamil Říha, Ph.D.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma „Automatická kontrola správnosti sestavní výrobku“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

Poděkování

Děkuji vedoucímu práce Ing. Kamilovi Říhovi, Ph.D. za velmi užitečnou metodickou pomoc a cenné rady při zpracování této diplomové práce. Rovněž děkuji Ing. Pavlovi Kaniovi ze společnosti Honeywell za poskytnutí potřebných informací, podpory a užitečných rad.

V Brně dne ....

podpis autora

Obsah

Úvod

Nedokonalá výstupní kontrola při výrobě digitálních termostatů ve společnosti Honeywell přivedla pracovníky vývojového centra téže společnosti na myšlenku využít metod zpracování digitálních obrazů pro bezkontaktní výstupní kontrolu sestavení výrobku. Cílem této práce je prozkoumat metody zpracování obrazu a navrhnout algoritmus, který bude kontrolovat a klasifikovat různorodé vlastnosti výrobků – barvu, tvar, typ loga, přítomnost některých prvků.

V úvodní části je popsána motivace, která vzniku práce předcházela. Po stručném popisu sledovaných vlastností výrobku následuje teoretický rozbor dvou metod, ze kterých autor vycházel při sestavování výsledného algoritmu. Dále přichází s vlastním návrhem metody pro kompenzaci nerovnoměrného osvětlení ve scéně, která je založena na modelování jasových ploch pomocí Bézierových křivek. Ke každé metodě jsou naznačeny alternativní postupy doplněné o zdůvodnění výběru jednotlivých metod.

Při hledání vhodných metod, byl důraz kladen i na časovou náročnost algoritmů a z tohoto důvodu jsou nastíněny i prostředky, které může programátor v prostředí Windows využít k měření časových intervalů.

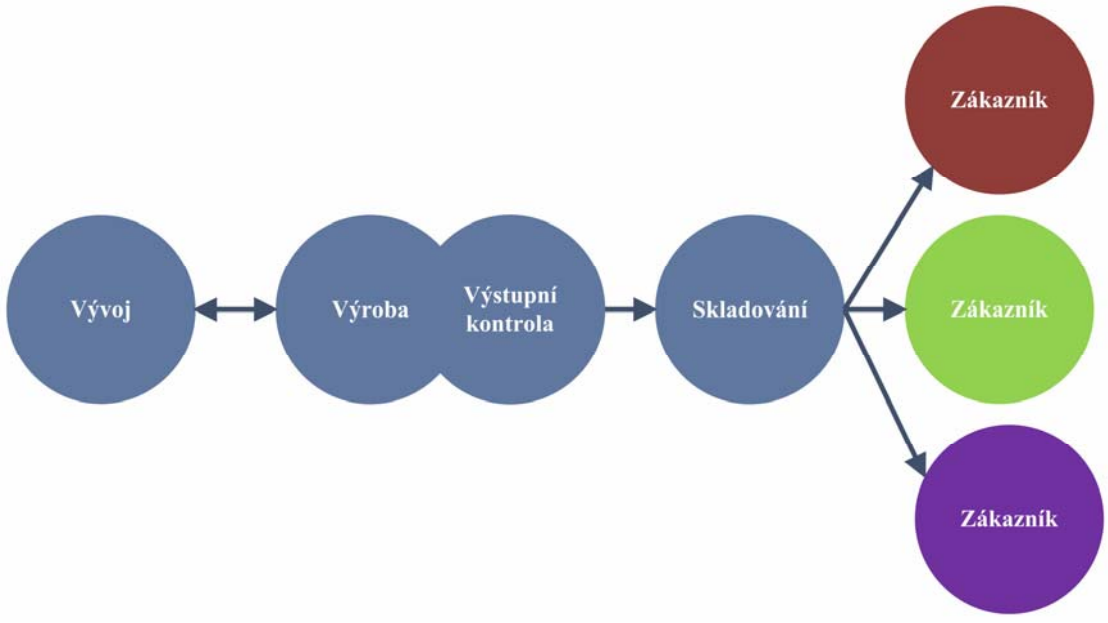
Pro vlastní realizaci je zvolen jazyk C/C++. Jedním z požadavků zadavatele bylo využití tzv. svobodného software. Proto je krátká stať věnována právním aspektům licence BSD.

1 Současný stav problematiky

Předmětem této práce je návrh softwarového řešení systému pro bezkontaktní kontrolu správnosti sestavení výrobku. Oním kontrolovaným výrobkem je moderní digitální termostat z produkce fy. Honeywell (nadále je použito pouze označení termostat). Níže je stručně vysvětlena motivace, která předcházela vzniku této práce. Dále jsou čtenáři představeny důležité parametry termostatu, které kontrolní algoritmus vyhodnocuje.

1.1 Potřeba automatizované kontroly

Proces vývoje a výroby tak jak je schématicky zobrazen na obr. 1.1 je zcela intuitivní a netřeba jej nikterak více specifikovat. Z důvodu zachování názornosti nejsou zobrazeny zpětné vazby od zákazníků. Navržený kontrolní algoritmus spadá do bloku Výstupní kontrola, který je přímo navázán na proces výroby. Na první pohled je patrná důležitost tohoto procesu. Jedná se o poslední místo, kde lze zachytit vadné výrobky před jejich dodávkou zákazníkovi. Přistupme nyní k popisu konkrétního výrobního procesu termostatu.



Obr. 1.1: Proces vývoje a výroby.

Termostat zobrazený na obrázku 1.2 je sestavován z několika různých částí. Výsledný produkt se liší v mnoha vlastnostech, které lze členit do dvou skupin. Předně jde o vlastnosti funkční, které determinuje vnitřní elektronika - způsob napájení, varianta komunikačního rozhraní, firmware, hardware, display a další. Druhou skupinu tvoří vlastnosti popisující plastový kryt výrobku, tedy barva, velikost, varianta montážních prvků, logo, atd.

V zásadě platí, že určité množině funkčních vlastností lze přiřadit jednoznačnou kombinaci vlastností krytu. Naopak jednoznačné přiřazení neplatí (např. různé varianty firmware mohou sdílet jedinou variantu krytu).

Obr. 1.2: Sestavený termostat připravený ke kontrole.

Stávající proces výstupní kontroly obsahuje kontrolní mechanismy pouze pro funkční vlastnosti výrobku. Druhá množina vlastností (varianty krytu) není v procesu nikterak kontrolována. Cílem této práce je navrhnout takový algoritmus digitálního zpracování obrazu, který by uvedený procesní nedostatek odstranil. Nutnost zavedení kontroly v tomto místě posiluje fakt, že do konečné podoby je termostat kompletován manuálně (vyšší riziko chyb oproti automatizované kompletaci).

Nejběžnější chybou je osazení vnitřní elektroniky nesprávnou variantou krytu. Jednotlivé varianty termostatu jsou určeny různým zákazníkům a jakákoliv chybovost je

tedy nepřijatelná. Je nemyslitelné, aby například zákazník A obdržel dodávku termostatů se správnou elektronikou, nicméně s krytem zákazníka B.

Tuto práci motivuje snaha o celkové zlepšení a zvýšení stupně automatice výrobního procesu termostatu a dále možné snížení logistických nákladů spojených s případnými zákaznickými reklamacemi.

1.2 Kryty termostatu

Následuje kompletní výčet vlastností jednotlivých variant krytů. Každá z nich vznáší konkrétní požadavky na detekční algoritmus, tyto jsou níže rovněž stručně diskutovány.

V tabulce 1.1 jsou uvedeny přípustné barevné variace termostatických krytů. V současné době (duben 2010) jsou definovány pouze tři barvy, přičemž nejsou k dispozici jejich přesné souřadnice v žádném z barevných modelů. Je vhodné v tomto ohledu sestavit detekční algoritmus dostatečně univerzálně tak, aby v budoucnu nebylo problematické dodefinování další barevné varianty.

Problematika měření barevnosti snímané scény je úzce spjata s metodami vyvážení bílé barvy a s kompenzačními technikami nerovnoměrného osvětlení, které společně zajišťují konstantní barevné podání snímané scény při proměnném osvětlení. Konkrétní řešení tohoto problému je představeno v kapitole 2.2.

Tab. 1.1: Přehled barevných variant krytů (odstíny barev jsou pouze orientační).

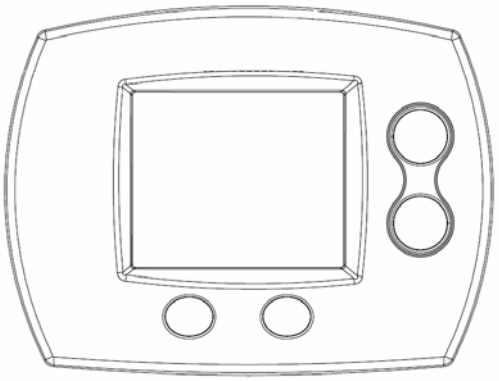
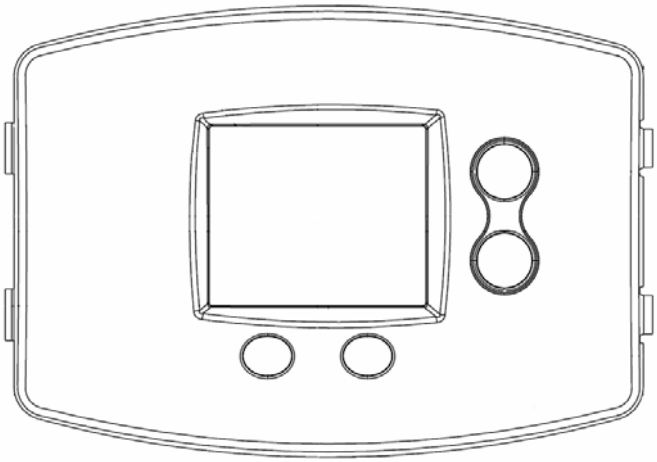
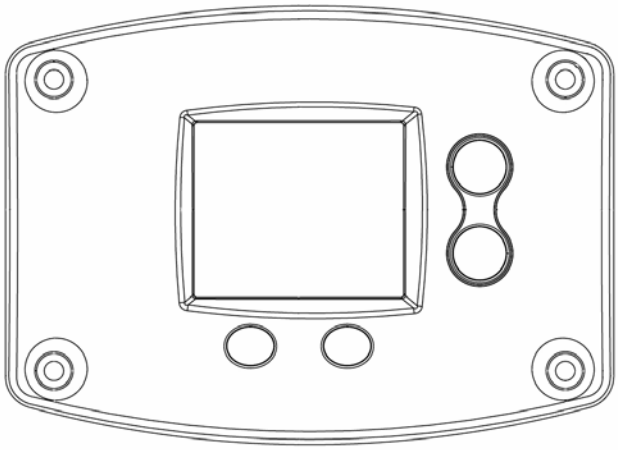
Bílá	
Stříbrná	
Modrá	

Tabulka 1.2 přináší přehled typizovaných krytů včetně jejich vnějších rozměrů. Za povšimnutí stojí, že kryty se neliší pouze tvarem, ale i druhem montážních prvků. Tyto prvky jsou klíčové pro rozlišení variant 2 a 3. Rozpoznání jednotlivých variant se ve zpracování obrazu převádí na triviální úlohu detekce hran.

Posledním významným prvkem na každém krytu je zákaznické logo. Toto je vždy umístěno v prostoru nad displejem a jeho orientační rozměry jsou 30 x 10 milimetrů. Všechna v současné době (duben 2010) známá loga jsou uvedena v přehledové

tabulce 1.3. Klasifikace tohoto prvku klade nejvyšší nároky na použitý převodník obrazových dat do digitální podoby a to jak po stránce rozlišení pořízeného obrazu, tak po stránce míry šumu v obrazu.

Tab. 1.2: Přehled definovaných typů krytů a jejich vnější rozměry v milimetrech (šířka x výška).

Typ 1	
	114,3 x 86,4
Typ 2	
	147,7 x 103,3
Typ 3	
	143,5 x 103,3

Tab. 1.3: Varianty zákaznických popisek.

Logo 1	
Logo 2	
Logo 3	
Logo 4	
Logo 5	
Logo 6	
Logo 7	
Logo 8	Vystouplý nápis Honeywell
Logo 9	Prázdné místo bez loga

2 Vybrané metody zpracování obrazu

Další text pojmenovává některé konkrétní úkoly systému pro kontrolu sestavení termostatu a naznačuje metody, které umožní tyto úkoly řešit. Problematika digitálního zpracování obrazových dat je velice široká a pro lepší čtenářovu orientaci je tedy vhodné zavést alespoň nejzákladnější klasifikaci a terminologii, která umožní rozklad složité komplexní úlohy na posloupnost dílčích méně náročných úkolů.

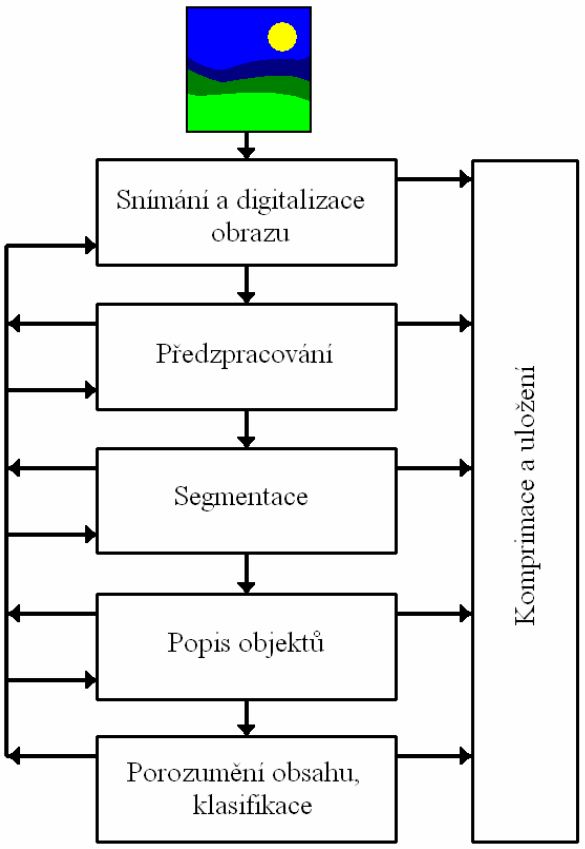
2.1 Úvod do zpracování obrazu

Poněkud upravená klasifikace metod zpracování obrazu od autorů [1] umožňuje zavedení následujících kategorií:

- **Snímání a digitalizace obrazu**. Přes optickou soustavu obrazového AD převodníku je snímaná scéna převedena do digitální podoby. Tento proces je tak jako každý jiný AD převod doprovázený ztrátou informace vlivem konečného rozlišení převodníku v horizontálním i vertikálním směru a kvantováním jasu každého obrazového bodu na omezený počet úrovní. Optická soustava předřazená AD převodníku mimo jiné často plní úlohu antialiasingového filtru.
- **Předzpracování** se provádí za účelem zvýraznění některých rysů v obraze obsažených. Patří sem například filtrace obrazu (rozmazání, zvýraznění hran, filtrace antialiasingovou dolní propustí předcházející podvzorkování obrazu, potlačení šumu), jasové transformace (ekvalizace histogramu, gama korekce), geometrické transformace (posunutí, rotace, zkosení, perspektiva,...), morfologické operace (dilatace, eroze, ...), nebo převody obrazových dat mezi různými barevnými modely. V kapitole 2.2 je představena metoda kompenzace nerovnoměrného osvětlení scény, která rovněž spadá do této kategorie.
- **Segmentace** se provádí s cílem rozdělit prostý obraz na objekty reálného světa. Na této úrovni se využívá metod prahování, detekce hranic objektů. Někteří autoři, např. [2], do této kategorie řadí i hranové detektory. Ve své podstatě se ovšem jedná o speciální případ filtrace obrazu a proto jsou pro účely tohoto textu zařazeny do kategorie předzpracování. V rámci kontrolního systému lze identifikovat úlohu lokalizace termostatu ve scéně, která spadá právě do kategorie segmentace obrazu. Blíže je o této problematice pojednáno v kapitolách 2.3, 2.4.
- Dalším krokem je **popisu** nalezených **objektů** pomocí definovaných příznaků.

- **Porozumění obsahu obrazu** se často zužuje na prostou **klasifikaci** nalezených a příznaky popsaných **objektů**.
- Samostatnou skupinu utvářejí algoritmy a metody zabývající se **kompresí obrazových dat** a jejich **uchováním** na paměťovém médiu.

Zavedená klasifikace je graficky zobrazena na obrázku 2.1. Důležitý je především způsob jakým na sebe jednotlivé úrovně zpracování navazují, jak se vzájemně ovlivňují. Podstatná je rovněž i zpětnovazební síť, která umožňuje vyšším vrstvám adaptivně parametrizovat vrstvy nižší. Konkrétní aplikace nemusí obsahovat všechny uvedené úrovně, jejich volba je primárně určena cílem zpracování.



Obr. 2.1: Základní klasifikace metod zpracování obrazu.

2.2 Kompenzace osvětlení

Korektní podání barev ve scéně vystavené proměnlivému zdroji osvětlení zajišťují algoritmy vyvážení bílé barvy. Konkrétní metoda je uvedena například v [3]. Je důležité si uvědomit, že obrazová data, která získáme nejběžnějším způsobem jako výstup z kamery či fotoaparátu jsou nevratně zkomprimována některým z kompresních algoritmů. Část informací, které jsou podstatné pro správné vyvážení bílé, je nenávratně ztraceno a tím jsou výrazně omezeny možnosti korekčních algoritmů. Implementace vyvážení bílé nad takovými daty přesahuje rozsah této práce. Vzhledem k povaze obrazových dat (bez redundantních informací), lze přepokládat ne příliš efektivní výsledky.

Dalším nepříjemným jevem v souvislosti s měřením barev je nerovnoměrné rozložení osvětlení ve snímané scéně, které vyvolá bodový zdroj světla na rovinné ploše. Tento jev je zcela intuitivní a můžeme jej pozorovat běžně v každodenním životě. Lidský mozek jej dokážeme předvídat a popsat i bez bližší znalosti jakýchkoliv fyzikálních veličin. Tato kapitola je motivována snahou o zajištění konstantních světelných podmínek ve snímané scéně. Na úvod je velmi stručně přiblížena ustálená terminologie z oblasti fotometrie.

2.2.1. Intenzita osvětlení a jas

Při zpracování této kapitoly bylo využito skriptum [4].

Nechť bodový zdroj osvětlení je popsán svojí polohou v prostoru a svítivostí I (cd). Pro potřeby tohoto textu není nezbytné tuto vlastnost blíže specifikovat, případný zájemce najde definici v [4]. Pakliže energie elektromagnetického záření klesá se čtvercem vzdálenosti r od bodového zdroje, potom intenzita osvětlení E vyvolaná zdrojem o svítivosti I rovněž klesá se čtvercem vzdálenosti r. Tuto závislost lze popsat vztahem

$$E = \frac{I}{r^2}. \qquad \text{(lx)} \qquad (2.1)$$

Dopadají-li paprsky z bodového zdroje A na plochu takovým způsobem, že svírají s normálou plochy úhel α, můžeme pro intenzitu osvětlení plochy psát

$$E = \frac{I}{r^2} \cdot \cos\alpha \,. \qquad \text{(lx)} \qquad (2.2)$$

Z hlediska lidského vnímání světla je nejvýznamnější fotometrickou veličinou jas. Jas lze vyjádřit jako poměr svítivosti dI, kterou disponuje svítící, nebo odrážející elementární ploška dS ve směru k pozorovateli B, ku průmětné ploše této elementární plošky na směr pozorování

$$L = \frac{\mathrm{d}I}{\mathrm{d}S \cdot \cos\beta}, \qquad \text{(cd·m}^{-2}\text{)} \qquad (2.3)$$

kde β je úhel mezi normálou plochy a směrem pozorování.

Osvětlením povrchu se každá jeho elementární ploška sama stává zdrojem osvětlení. Parametry takového zdroje závisí především na vlastnostech osvětlovaného povrchu. Reakce povrchu na dopadající osvětlení může být velmi různorodá, známe povrchy, které odrážejí světelné paprsky dle zákona odrazu, jiné světlo téměř dokonale pohltí. Představme si povrh dokonale rozptylující do všech možných směrů stejnou měrou a popišme jej parametrem ρ, který nazvěme odrazivost. Pro jas této plochy lze odvodit

$$L = \frac{E \cdot \rho}{\pi}, \qquad \text{(cd·m}^{-2}\text{)} \qquad (2.4)$$

kde E je intenzita osvětlení plochy bodovým zdrojem.

Jas osvětlené plochy dle (2.4) je obrazovým AD převodníkem digitalizován a kvantován určitou blíže nespecifikovanou převodní funkcí f(.) do jasu L_A z rozsahu <0; 2^n-1>, kde n je počet bitů použitých pro reprezentaci jednoho obrazového elementu. Předchozí můžeme zapsat jako

$$L_A(x,y) = f(L). \qquad \text{(-)} \qquad (2.5)$$

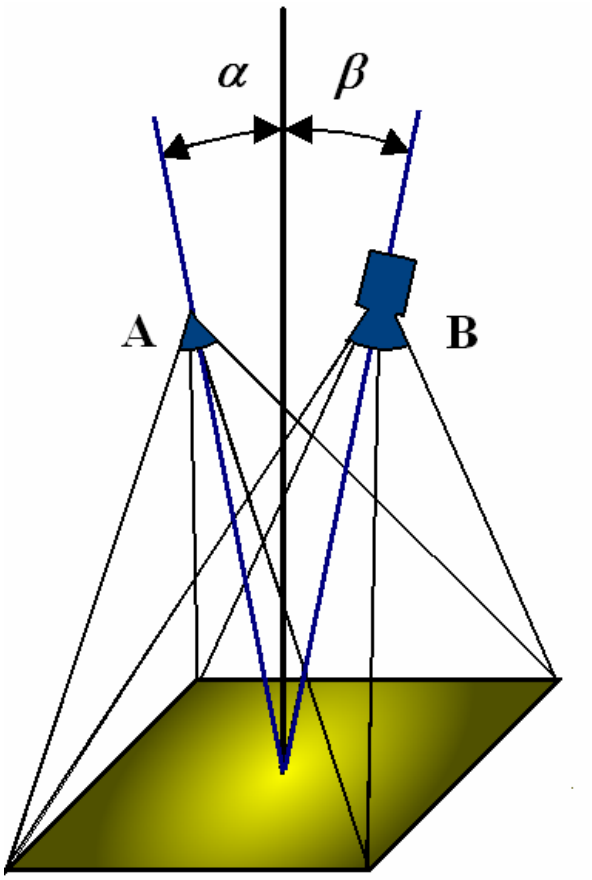
Proměnné x,y naznačují, že L_A je diskrétní funkcí prostorových souřadnic.

Úkolem kompenzačního algoritmu je zabezpečit konstantní odezvu snímané homogenní scény při jejím osvětlení bodovým zdrojem. Základním předpokladem je

umístění bodového zdroje A i pozorovatele (kamery) B v blízkosti normály osvětlované, resp. pozorované scény (obr. 2.2). Pro úhly blízké nule, lze ve vztazích (2.2), (2.3) zavést zjednodušení

$$\begin{aligned} \cos(\alpha) &\cong 1 \\ \cos(\beta) &\cong 1 \end{aligned} \quad . \qquad (-) \qquad (2.6)$$

Z rovnic (2.2) a (2.4) po zjednodušení (2.6) vyplývá, že maximální jasový gradient ve snímané scéně se odvíjí od kvadrátu vzdálenosti bodového zdroje r^2. Této vlastnosti je využito v dalším postupu.

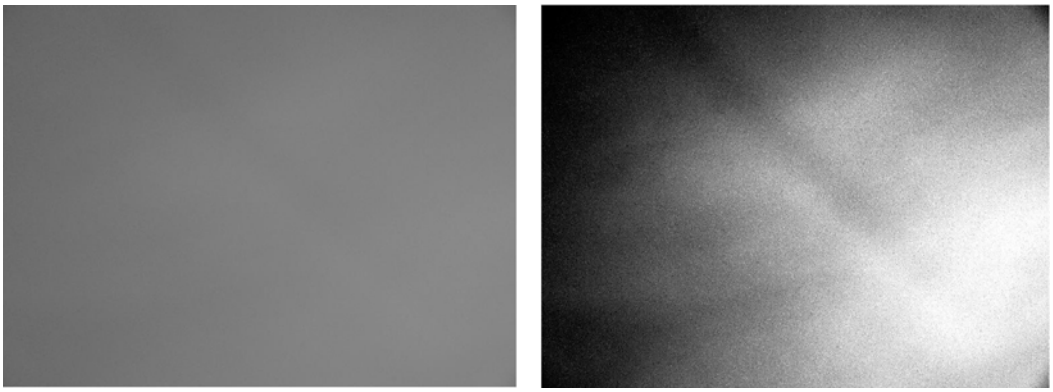


Obr. 2.2: Osvětlení a snímání scény.

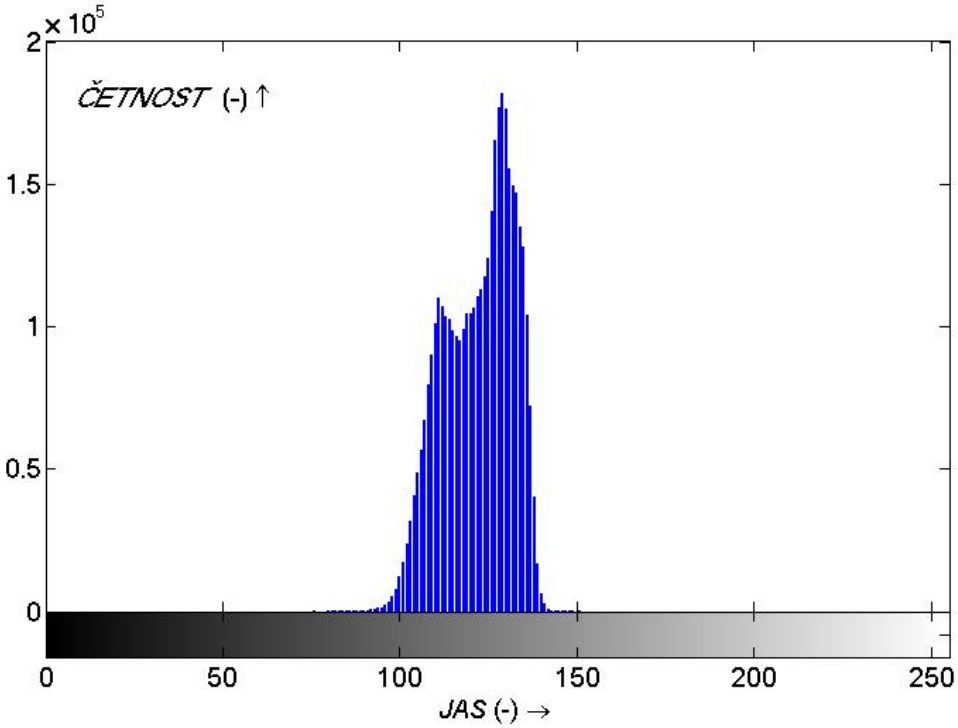
2.2.2. Jas jako plocha

Obr. 2.3 vlevo zobrazuje šedotónovou fotografii bílého listu papíru, který byl vystaven působení bodového zdroje osvětlení. Na první pohled se obraz jeví jako homogenní. Při pohledu na jeho histogram (obr. 2.4) je však patrné, že obraz příliš homogenní není.

Nehomogenitu uvedeného obrazu lze zvýraznit ekvalizací histogramu, takto upravený snímek je zobrazen na obr. 2.3 vpravo.



Obr. 2.3: Vlevo nerovnoměrně osvětlený bílý papír. Vpravo stejný obraz po ekvalizaci.

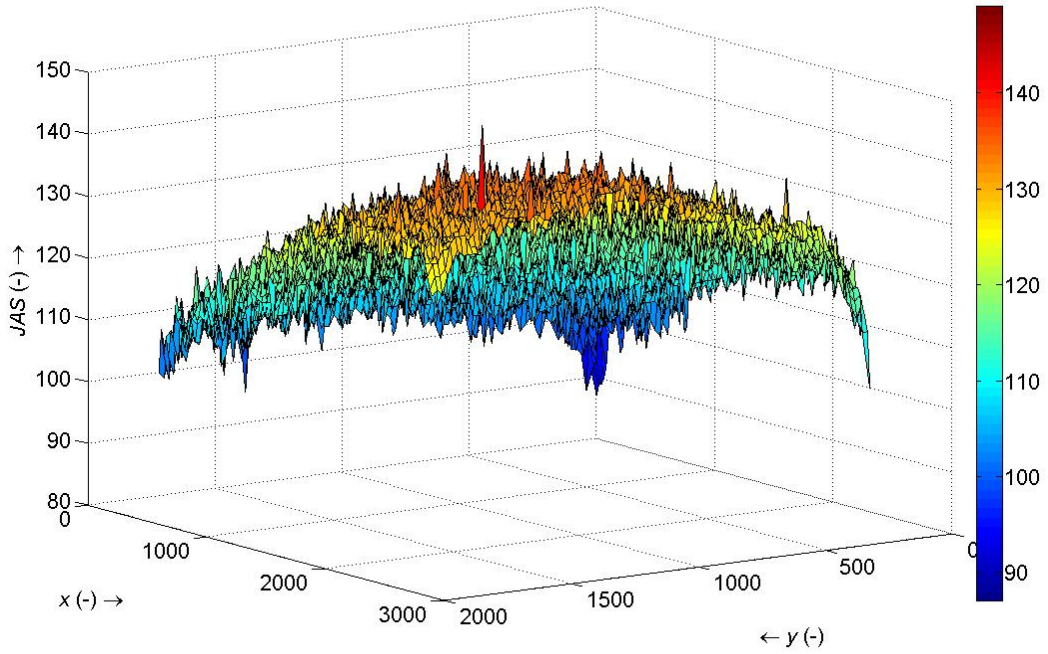


Obr. 2.4 Histogram obrazu 2.3 vlevo.

Úroveň jasu ve vyfotografované scéně lze interpretovat jako trojrozměrnou plochu. Jinými slovy, hodnota jasu každého pixelu moduluje výšku plochy na dané souřadnici. Takto byla z obr. 2.3 vlevo odvozena plocha na obr. 2.5. Lze rozpoznat určitý kopulovitý tvar plochy porušený šumem. Tento tvar je projevem nerovnoměrného

osvětlení ve snímané scéně. Ideálně osvětlená scéna by v této interpretaci byla zobrazena jako hladká plocha rovnoběžná s rovinou *xy*.

Sečtením plochy popisující jas v obraze s přesně inverzní plochou dojde v ideálním případě k odstranění uvedeného nedostatku. Matematická formulace inverzní plochy je obsahem následujících kapitol. Pro modelování těchto ploch se jeví jako optimální Bézierova a B-spline matematika.



Obr. 2.5: Interpretace obrazu 2.3 vlevo jako plochy v prostoru.

2.2.3. Bézierovy plochy

Problematika modelování ploch (kapitoly 2.2.3 až 2.2.5) je zpracována na základě kapitoly 5 knihy [5].

Modelování obvykle probíhá tak, že je definováno několik řídících bodů a matematický aparát z jejich polohy určí tvar celé plochy. Plochy použité v tomto textu obecně řídící body neprotínají, pouze je aproximují. Chybné zadání řídícího bodu poté nedeformuje plochu to takové míry jako v případě použití interpolačních ploch.

Bézierova plocha n x m-tého stupně je určena $(n + 1)$ x $(m + 1)$ řídícími body $P_{i,j}$ a vztahem

$$Q(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}P_{i,j}B_i^n(u)B_j^m(v). \tag{2.7}$$

Takovou plochu si lze představit jakou soustavu polynomiálních křivek v horizontálním a vertikálním směru [5]. Celkový počet řídících bodů je $n \cdot m$.

Bernsteinovy bázové polynomy k-tého stupně definují Beziérovy křivky dle rovnice

$$B_\upsilon^k(t)=\binom{k}{\upsilon}t^\upsilon(1-t)^{k-\upsilon}, \tag{2.8}$$

kde $\nu = 0, ..., k$. Pro polynomy stupně k lze z předchozího odvodit $(k + 1)$ polynomů. Konkrétně pro rovnici (2.7) lze z (2.8) odvodit bázové křivky v obou směrech(horizontálním i vertikálním) jako

$$\begin{aligned} B_i^n(u)&=\binom{n}{i}u^i(1-u)^{n-i},\\ B_j^m(v)&=\binom{m}{j}v^j(1-v)^{m-j}. \end{aligned} \tag{2.9}$$

Dosazením hodnot $Q(0,0)$, $Q(1,0)$, $Q(0,1)$ a $Q(1,1)$ lze ověřit, že rohy Bézierovy plochy jsou identické s body P_{00}, P_{10}, P_{01} a P_{11} řídící sítě. Následuje přehled některých dalších důležitých vlastnosti Bézierových ploch. Celá plocha leží v konvexní obálce svých řídících bodů. V případě změny jednoho řídícího bodu mění svůj tvar celá plocha. Tato vlastnost často vede k dělení větších ploch na menší, tzv. pláty, každý z nich se potom modeluje samostatně. S tímto je spojen nárůst počtu řídících bodů, ovšem případná změna některého z nich způsobuje pouze lokální ovlivnění okolních ploch a není tak nutné modelovat celou rozlehlou plochu znovu.

Při napojování Bézierových plátů se snažíme dosáhnout přirozeného přechodu jedné plochy ve druhou, tak aby napojení působilo přirozeně a nebylo viditelné. Hladkost tohoto spoje lze matematicky popsat a nazýváme ji spojitostí. Rozeznáváme spojitost parametrickou C a geometrickou G. Příklady nejběžnějších spojitostí dle [5] jsou:

- Pláty jsou C^0 spojité pokud je jejich společná strana alespoň třídy C^0 (alespoň spojitá). Jinými slovy je nutné ztotožnit řídící body, které určují příslušnou stranu. Při tomto způsobu napojování mohou vznikat ostré hrany.
- Pláty jsou C^1 spojité pokud je jejich společná hrana alespoň třídy C^1 (spojitost první derivace) a navíc všechny příčné parciální derivace v jejích bodech jsou identické pro napojované pláty (provázání čtyř dvojic řídících bodů).
- Pláty jsou G^1 spojité pokud je jejich společná hrana alespoň třídy G^1, příčné derivace nemusí být identické, stačí, když jsou pouze lineárně závislé. Tato podmínka není tolik omezující jako u C^1 a umožňuje větší manipulaci s řídícími body. Obecně platí, že C^n implikuje C^{n-1} a C^n implikuje G^n.

Požadavek hladkého navazování spolu svazuje řídící body sousedních plátů a tím omezuje variabilitu celé plochy.

2.2.4. Bézierovy bi-kubiky

V části 2.2.1 je odvozeno, že maximální jasový gradient ve snímané scéně je dán polynomem druhého řádu. Modelování kompenzační plochy s dostatečnou tvarovou variabilitou zajistí polynomy s řádem o jedna vyšším. Polynomiální křivka třetího řádu je nazývána kubikou, plochu modelovanou z těchto křivek tak lze nazvat bi-kubikou.

Dosazením řádu polynomu $n = m = 3$ do (2.7) je odvozen obecný předpis pro Bézierovy bi-kubiky ve tvaru

$$Q(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} P_{i,j} B_i^3(u) B_j^3(v). \tag{2.10}$$

Takovou plochu plně popisuje jejích 16 řídících bodů.

Bernsteinovy polynomy třetího stupně definuje předpis (dosazením do (2.8))

$$B_v^3(t) = \binom{3}{v} t^v (1-t)^{3-v}, \tag{2.11}$$

resp. jejich výčet

$$
\begin{aligned}
B_0^3(t) &= (1-t)^3 \\
B_1^3(t) &= 3t(1-t)^2 \\
B_2^3(t) &= 3t^2(1-t) \\
B_3^3(t) &= t^3
\end{aligned}
\qquad (2.12)
$$

Dosadit z (2.12) do (2.10) lze záměnou parametru t za u, resp. v.

Nyní lze přistoupit k odvození maticového zápisu rovnice (2.10), který může být přímo implementován do výpočetního prostředku. Bernsteinovy polynomy (2.12) mohou být zapsány jako součin matic

$$
\mathbf{B} = \mathbf{TM} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \qquad (2.13)
$$

Konečný předpis lze odvodit dosazením (2.13) do (2.10) a rozložením popisu plochy do prostorových souřadnic x, y, z jako

$$
\begin{aligned}
x(u,v) &= \mathbf{UMP}_x\mathbf{M}^{\mathrm{T}}\mathbf{V}^{\mathrm{T}} \\
y(u,v) &= \mathbf{UMP}_y\mathbf{M}^{\mathrm{T}}\mathbf{V}^{\mathrm{T}} \\
z(u,v) &= \mathbf{UMP}_z\mathbf{M}^{\mathrm{T}}\mathbf{V}^{\mathrm{T}}
\end{aligned}
\qquad (2.14)
$$

kde $\mathbf{P}_x$, $\mathbf{P}_y$ a $\mathbf{P}_z$ jsou matice 4 x 4 normalizovaných souřadnic řídících bodů, vektory $\mathbf{U}$, $\mathbf{V}$ jsou odvozeny z vektoru $\mathbf{T}$ prostou záměnou parametru jako $\mathbf{U} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$, resp $\mathbf{V} = \begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix}$.

Dosazování za u, v z rozsahu $\langle 0;1 \rangle$ do (2.14) generuje plochu

$$
Q(u,v) = [x(u,v), y(u,v), z(u,v)]. \qquad (2.15)
$$

Je-li při modelování zajištěno rovnoměrné rozložení řídících bodů v celé ploše, není nutné počítat složky $x(u,v)$, $y(u,v)$. Celý výše popisovaný aparát se v takovém případě redukuje na prosté přiřazení

$$\begin{aligned} x(u,v) &= v, \\ y(u,v) &= u. \end{aligned} \tag{2.16}$$

Implementace je velice jednoduchá a spočívá v cyklickém výpočtu třetí rovnice z (2.14) pro každý obrazový bod, výsledkem je *z*-ová složka plochy v příslušném bodě. Příklad implementace v prostředí Matlab je uveden v kapitole 3.6.1. Některé další specifika implementace v jazyku C/C++ jsou diskutována v kapitole 3.6.2.

2.2.5. B–spline plochy

Alternativu k Bézierovým bi-kubikám představují B-spline plochy. V zásadě se jedná o totožný matematický aparát s několika málo odlišnostmi. Tvar plochy je opět odvozen od sítě řídících bodů. Oproti Bézierovým plátům se navazující B-spline plát definuje použitím *m* x (*n* - 1) bodů plátu předchozího, přidá se pouze *m* nových řídících bodů. Takto dochází k překrývání řídících sítí jednotlivých plátů. V případě bi-kubik mají sousední pláty společné vždy tři sloupce nebo tři řádky řídících bodů.

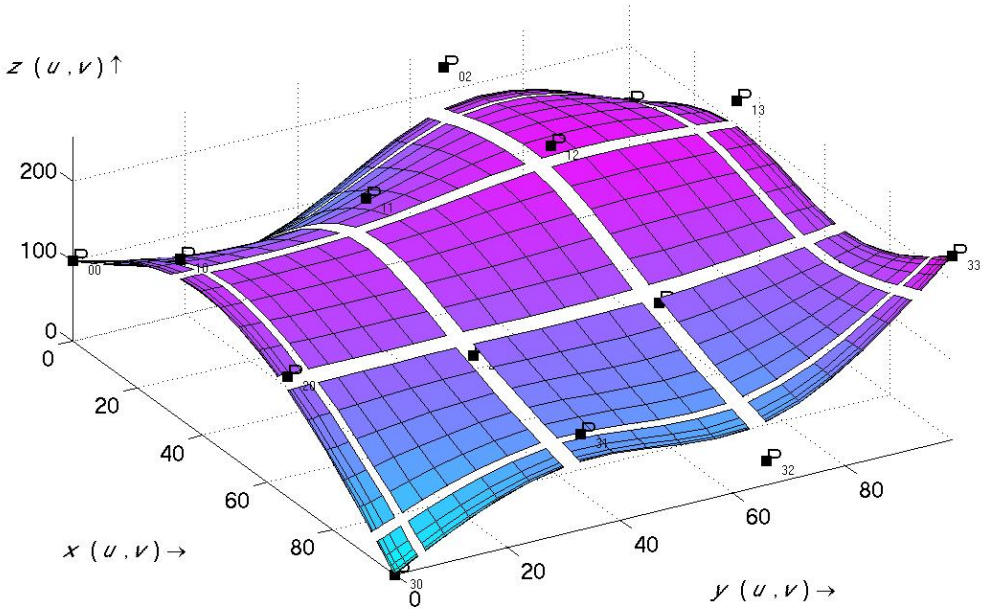
V důsledku těchto vlastností se B-spline (Coonsovy) pláty velmi snadno navazují. B–spline plochy stupně *n* zaručují C^{n-1} spojitost bez nutnosti omezování řídících bodů vnějšími podmínkami, tak jako tomu je u Bézierových ploch. Změna jediného řídícího bodu ovlivňuje více plátů, to však není na obtíž. Podstatné je, že takto vyvolaná změna je lokální a neovlivňuje plochu jako celek. Tyto výhodné vlastnosti, jsou vykoupeny některými specifiky při generování jednotlivých plátů. Stejně jako u Bézierových ploch i Coonsova plocha leží celá v konvexní obálce svých řídících bodů. Plocha obecně neprochází krajními body řídící sítě, toho lze docílit násobností řídících bodů, viz. [5].

Generující polynomy vycházejí z Coonsových kubik. Bázová matice má tvar

$$\mathbf{M} = \frac{1}{6} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}. \tag{2.17}$$

Generování jednotlivých složek plochy (2.15) probíhá stejným způsobem dle (2.14). Pro Coonsovu plochu je nutné vypočítat všechny tři dílčí složky $x(u,v)$, $y(u,v)$ a $z(u,v)$. Tato skutečnost představuje zásadní obtíž pro implementaci Coonsovy plochy nad pevně daným rastrem souřadnic *uv*. Vygenerované souřadnice $x(u,v)$ a $y(u,v)$ obecně neodpovídají původním souřadnicím *uv* a neplatí tedy přiřazení (2.16).

Aby Coonsova bi-kubika pokryla celou řídící síť 4 x 4 řídících bodů, musí být vygenerováno 5 x 5 plátů (v případě Bézierovy bi-kubiky stačí plát jediný). Rozložení plátů ve výsledné ploše naznačuje obrázek 2.10. Ke konstrukci okrajových plátů se využívají tzv. dvojné a trojné řídící body. Blíže je celý postup popsán v kapitole 5 knihy [5], na základě této knihy byla zpracována i celá tato kapitola.



Obr. 2.6: Plocha složená z 25ti Coonsových plátů.

2.3 Houghova transformace

Houghova [hafova] transformace (dále jen HT) patří mezi techniky, které se zabývají segmentací obrazu. Do stejné skupiny patří například i metody prahování. Zásadní odlišnost HT spočívá v tom, že poskytuje globální pohled na celý obraz, zabývá se vzájemnými vztahy mezi pixely vstupního obrazu. Metody prahování naopak pracují pouze lokálně, bez znalosti širšího okolí. HT řeší úlohu lokalizace objektů ve snímané scéně, určení jejich počtu, velikosti či orientace.

Klasická HT, tak jak ji představil v roce 1962 Paul Hough, hledá v obraze přímky. Později byla HT zobecněna pro vyhledávání obecných tvarů, nejčastěji kružnic, elips a jiných analyticky popsatelných útvarů [2].

Detekční algoritmus využívá HT k lokalizaci kulatých tlačítek umístěných v pravé polovině krytu termostatu, viz. obr. 1.1. Motivace k této operaci je popsána v kapitole 3.7 spolu s popisem implementace HT v jazyce C/C++. Následuje popis principů HT obecně a popis HT pro detekci kružnic, která je konkrétně implementována v projektu.

2.3.1. Houghova transformace obecně

Na vstupu HT je obvykle předkládán binární obraz, který je získán prahováním výsledků hranové detekce v originálním snímku. HT ve zdrojovém obraze posuzuje vzájemný vztah pixelů, kvantifikuje míru s jakou se hrany blíží geometrickému útvaru, jenž je předmětem lokalizace. HT vychází z analytického zápisu hledaného útvaru. Analytický matematický zápis běžně obsahuje proměnné a parametry. Parametry určují charakter tzv. parametrického prostoru HT. Obecně se jedná o vícerozměrný prostor $\mathbf{R}^n$ jehož dimenze n je dána počtem parametrů v analytického předpisu. Například přímka má parametry dva (směrnice, posunutí přímky vůči počátku ve smyslu osy y) a proto i parametrický prostor bude dvourozměrný. Oproti tomu kružnice se svými třemi parametry (souřadnice středu, poloměr) vytvoří parametrický prostor dimenze 3.

Každý jeden bod zdrojového obrazu je transformován HT do parametrického prostoru, kde generuje nějaký geometrický útvar. Po transformaci všech bodů zdrojového obrazu vzniknou v parametrickém n-rozměrném prostoru lokální maxima. Souřadnice těchto maxim lokalizují a popisují hledaný útvar v rovině původních

obrazových souřadnic. Z podstaty věci plyne, že parametrický prostor lze implementovat ve výpočetním prostředku jako *n*-rozměrný akumulátor.

2.3.2. Houghova transformace kružnic

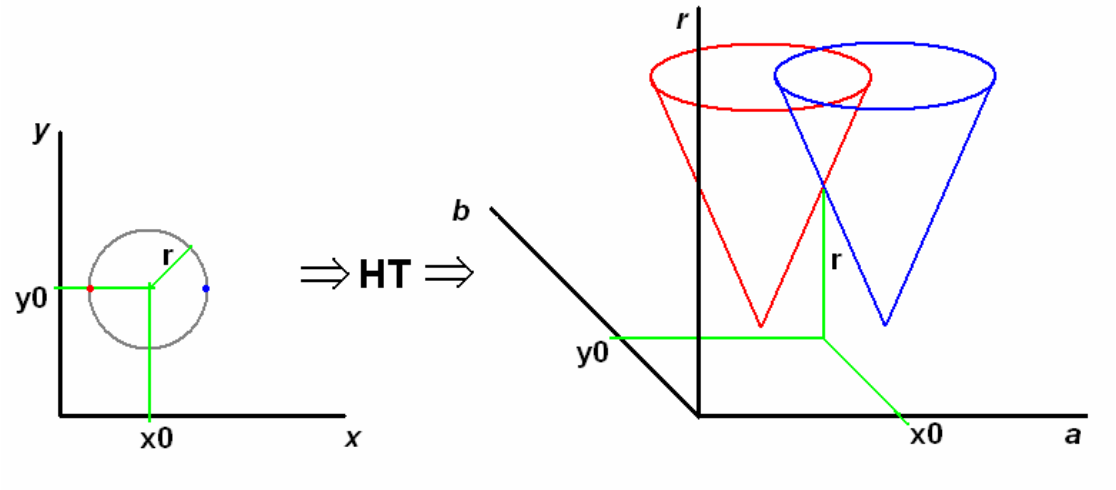
Následuje popis konkrétního příkladu HT modifikované pro detekci kružnic. Nechť osy kartézského souřadného systému jsou označeny x,y, potom kružnici v těchto souřadnicích lze popsat rovnicí

$$(x - x_0)^2 + (y - y_0)^2 = r^2, \tag{2.18}$$

kde bod (x_0,y_0) je středem kružnice a r je jejím poloměrem. Při konstrukci kružnice v kartézských souřadnicích jsou obvykle nejprve zvoleny parametry – střed (x_0,y_0) a poloměr r. Následuje volba proměnných x,y v takových kombinacích, aby byla splněna rovnost (2.18). Popsaným způsobem je bod po bodu zkonstruována celá kružnice.

Přechod z kartézských souřadnic do parametrického prostoru realizuje HT tak, že zamění roli proměnných a parametrů v rovnici (2.18). V parametrickém prostoru jsou souřadnice bodu (x,y) v roli parametrů a nemění se. Pro jedinečnou kombinaci parametrů jsou voleny kombinace proměnných x_0, y_0 a r takovým způsobem, aby platila rovnost (2.18). Výsledkem je kužel v parametrickém prostoru.

Leží-li dva body na kružnici, potom se jejich obrazy v parametrickém prostoru (dva kužely) střetávají v bodě o souřadnicích (x_0, y_0, r). Tyto souřadnice lokalizují a popisují velikost kružnice v původní obrazové rovině (obr. 2.11). Na obrázku je zároveň vidět, že oba kužely se střetávají ve více bodech. Z tohoto důvodu je nutné vygenerovat větší počet kuželů pro korektní určení jedinečných souřadnic (x_0, y_0, r). Zároveň existence většího množství průsečíků může způsobovat detekci falešných kružnic. Pro kvalitní vstupní obraz je průsečík s nevyšší četností dostatečně zřetelný a správně lokalizovaný.



Obr. 2.7: Znázornění Houghovy transformace kružnic. Vlevo dva body na kružnici v obrazové rovině. Vpravo obraz těchto dvou bodů v prametrickém prostoru.

Obrázek dále naznačuje jednu významnou nectnost HT – vysokou výpočetní náročnost. Výpočetní algoritmus musí projít celým vstupním obrazem pixel po pixelu (při základní implementaci dva vnořené cykly) a pro každý jeden pixel reprezentující hranu musí v parametrickém prostoru generovat trojrozměrný objekt (další tři vnořené cykly). Značnou část výpočetního výkonu dále spotřebovává prohledávání parametrického prostoru při hledání lokálních maxim, což samo o sobě představuje netriviální úlohu. Vysoké výpočetní nároky způsobují, že v reálných aplikacích se pracuje s maximálně parametrickými prostory $\mathbf{R}^3$.

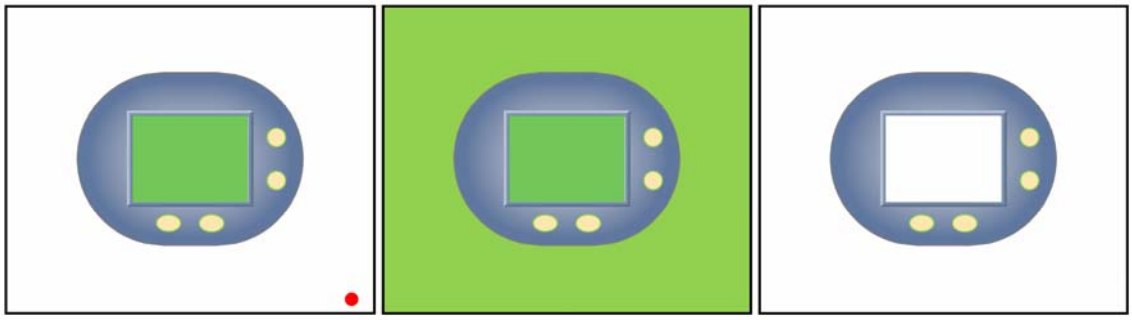
Výše uvedené skutečnosti vedou ke snahám o hledání optimálního algoritmu. V případě implementace pro vyhledávání kružnic představuje nejzákladnější optimalizační krok omezení přípustných poloměrů z rozmezí $0 \le r \le \infty$ do rozsahu $r_{MIN} \le r \le r_{MAX}$. U některých aplikací je možné se omezit pouze na jeden konkrétní poloměr r a v takovém případě dochází k redukci parametrického prostoru o jednu dimenzi.

2.4 Semínkové zaplavování

Semínkové zaplavování, v zahraniční literatuře označováno jako flood fill, seed fill někdy rovněž binary (grayscale) reconstruction [6], je metoda využívaná pro segmentaci obrazových dat na oblasti s rozdílnými vlastnostmi. Například pro oddělení pozadí od objektů v popředí snímané scény. Algoritmus lze rozdělit do dvou fází[6]:

- „Zasazení semínka" neboli identifikace pixelu v obrazu (seed point), který obsahuje specifickou informaci. Nemusí se nutně jednat pouze o jediný pixel, informace může být vyhodnocována z určitého okolí bodu, například mediánovým filtrem, vyhledáním maxima/minima, průměrné hodnoty a pod.
- Expanze semínka do okolních oblastí, které mají stejné nebo podobné vlastnosti jako seed point. Narazí-li algoritmus na pixel s odlišnými vlastnostmi, je expanze v tomto směru ukončena.

Po ukončení expanze jsou v obraze označeny všechny body, které leží ve spojité oblasti a mají shodné nebo podobné vlastnosti. Tímto způsobem dojde k oddělení oblastí obrazu s rozdílnými vlastnostmi. Rozdíl oproti metodám prahování spočívá v podmínce spojitosti označené oblasti. Rozdílnost obou přístupů je naznačena na následujícím obrázku.



Obr. 2.8: Porovnání výsledků algoritmů semínkového zaplavování a prahování

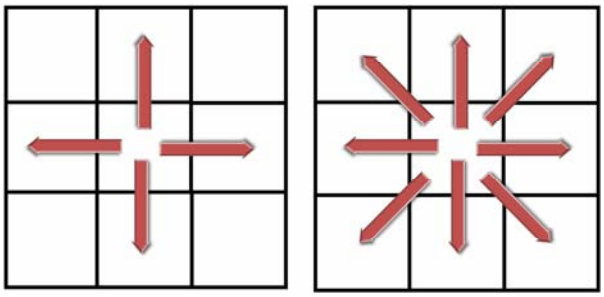
Prostřední obrázek zobrazuje originální snímek. Úkolem segmentace je oddělit spolehlivě termostat v popředí od pozadí. Důležitou vlastností snímku je, že barva displeje termostatu se blíží k barvě pozadí. Na obrázku vlevo je zobrazen výsledek algoritmu semínkového zaplavování, červeně je naznačen originální seed point. Z tohoto bodu algoritmus expanduje do okolí a pixely se shodnými hodnotami jako seed

point označuje bílou barvou. Je patrné, že dojde k označení celého pozadí a neoznačená oblast odpovídá přesně termostatu v popředí. Rovněž si lze povšimnout, že displej, který má podobné barevné vlastnosti jako pozadí, zůstal neoznačen. Nelze nalézt takovou cestu mezi seed pointem a oblastí displeje, kde by ležely pixely nesoucí podobnou barevnou informaci.

Na obrázku vpravo je zobrazen výsledek prahování originálního snímku. Práh je určen takovým způsobem, že pixely o hodnotách blízkých barvě pozadí jsou nastaveny do maximální saturace (na bílou barvu). Lze si povšimnout, že označena byla nejen oblast pozadí, ale i oblast displeje termostatu. Výsledkem je nedokonalá separace objektů ve scéně vlivem nevhodné volby segmentační metody.

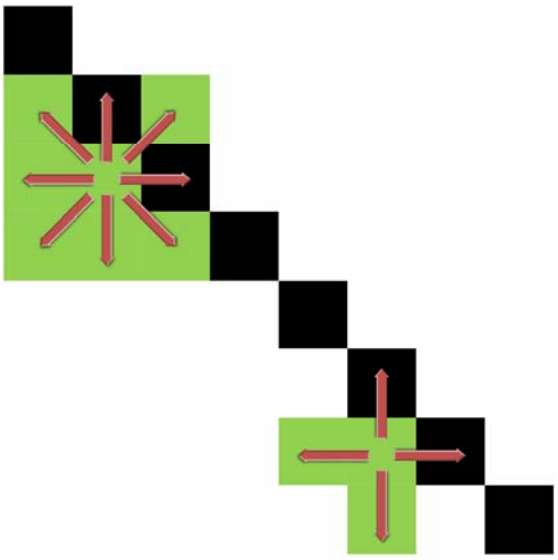
V případě, že se segmentace provádí prahovacími technikami je nutné zajistit, aby barvy jednotlivých oblastí byly jednoznačně a za každých okolností odlišné. Další možností je následné zpracování dat po segmentaci. Nabízí se například metody zpracování kontur [2], kdy je nutné v binárním obraze kontury identifikovat a následně vyhodnotit. Tento postup přináší vyšší nároky na algoritmizaci daného problému. Současně lze předpokládat, že implementace by byla časově méně výhodná, nežli v případě semínkového zaplavování.

Následující text krátce představuje základní varianty algoritmu [7]. Expanze ze seed pointu může probíhat ve 4-spojitém nebo v 8-spojitém rastru. V prvním případě dochází v obrazovém rastru k expanzi pouze po vodorovných a horizontálních liniích, viz. obr. 2.13 vlevo. Na obrázku vpravo jsou zobrazeny možnosti posunu v 8-spojitém rastru, oproti 4-spojitému rastru se přidávají diagonální posuvy.



Obr. 2.9: Čtyř-spojitý a osmi-spojitý rastr.

Pohyb v 8-mi spojitém rastru umožňuje překonat diagonální hrany o šířce jednoho pixelu, viz obr. 2.14. Hrana samotná samozřejmě zůstane neoznačena v obou případech.



Obr. 2.10: Chování algoritmu semínkového zaplavování na diagonální hraně.

Samotná implementace je možná rekurzivním i nerekurzivním přístupem [7]. V prvním případě funkce testuje body ve svém okolí (4-spojité/8-spojité) a rekurzí se volá z bodu, který byl označkován. V případě prohledávání rozlehlých obrazů hrozí přetečení hardwarového zásobníku. Druhou možností realizace je funkce s cyklem while a softwarovým zásobníkem. Do zásobníku jsou postupně vkládány body z okolí, které vyhoví kritériu. Po označkování a nové expanzi jsou ze zásobníku opět odebrány. Cyklus končí ve chvíli, kdy dojde k vyprázdnění zásobníku. Ačkoliv je v druhém případě zásobník implementován programátorem, je tato metoda efektivnější oproti hardwarovému zásobníku, jehož využití je spojeno s velkou režií při ukládání předávaných argumentů a návratových hodnot [8]. Programátor si sw. zásobník rovněž může opatřit kontrolními mechanismy, které zabrání jeho přetečení a tím pádu celého programu.

Kombinací výše uvedených přístupů lze obdržet následující 4 varianty algoritmu. Krátce jsou diskutovány i jejich základní vlastnosti [7].

- Rekurzivní metoda nad 4-spojitým okolím – nejjednodušší implementace z hlediska programátora, zároveň však nejpomalejší. Může způsobovat problémy s hw. zásobníkem při práci nad rozlehlými obrazy.

- Rekurzivní metoda nad 8-spojitým okolím – podobná předchozí variantě s tím rozdílem, že dokáže překonat diagonální hrany o tloušťce jednoho pixelu. Toto může být v některých aplikacích výhodou v jiných problémem. Díky této vlastnosti může být použita ke značkování tenkých linií či hran.
- Metoda se sw. zásobníkem nad 4-spojitým okolím – poskytuje možnost kontroly velikosti zásobníku a zabánit tak havárii programu. Výhodou je rovněž vyšší rychlost algoritmu oproti předchozím dvěma případům. Náročnější je implementace.
- Metoda se sw. zásobníkem nad 8-spojitým okolím – kombinuje zjevným způsobem vlastnosti předchozích dvou metod.

Společnou nectností výše popsaných metod je násobný přístup ke stejným obrazovým datům. Optimalizační kroky tedy směřují k tomu, aby každý bod v obraze byl testován pouze jedenkrát. Zpracovávají se například celé řádky či sloupce v obrazu apod. Podrobněji je tato problematika diskutována ve článcích [6, 7, 9]. Konkrétní implementace této metody je popsána v kapitole 3.7.

3 Implementace

Pro účely prvotního seznámení se s problematikou jsou některé dílčí funkce navrženy v prostředí Matlab s využitím modulu Image Processing Toolbox. Jde především o implementaci technik kompenzace nerovnoměrného osvětlení, o kterých pojednává kap. 2.2. Komplexní algoritmus je implementován v jazyce C/C++ v prostředí Microsoft Visual Studio, kde lze s výhodou využít open-source knihovnu pro počítačové vidění OpenCV (Open Source Computer Vision), dostupná na [10].

3.1 Knihovna OpenCV

Projekt OpenCV vznikl roku 1999 ve společnosti Intel, která přišla s myšlenkou otevřeného na platformě nezávislého prostředí pro podporu tvorby aplikací počítačového vidění pracujících v reálném čase. Pozornost vývojářů směřovala především k optimalizaci kódu. V roce 2006 vyšla verze 1.0 a následně v říjnu roku 2009 verze 2.0. Verze 2.0 inovuje především podporu pro více jádrové procesory [11].

Knihovna je napsána v jazyce C/C++ a v současné době (prosinec 2009) je dostupná ve verzích pro Linux, Windows a Mac OS X, připravují se rozšíření pro prostředí Python, Ruby, Matlab. OpenCV poskytuje nejen optimalizované metody a deklarace tříd pro zpracování obrazu, ale i nástroje pro uživatelskou interakci, metody pro ukládání obrazu do nejrozšířenějších formátů, implementace vybraných klasifikátorů, podporu pro snadné připojení kamery a některá další rozšíření. Celkem je v knihovně více než 500 metod [12].

Knihovna OpenCV, a to je pro nás podstatné, je svobodným software šířeným pod licencí BSD v upraveném znění z roku 1999, vzor k dispozici zde [13]. Problematiku této právní úpravy velmi podrobně diskutuje Josef Aujezdský ve svém článku [14]. Ve stručnosti uveďme pouze, že počítačový program opatřený tímto licenčním ujednáním je možné spojit s jiným počítačovým programem. Nabyvatel licence navíc není povinen poskytovat zdrojové kódy takového programu při jeho další distribuci [14].

Rovněž lze konstatovat, že knihovně OpenCV je poskytována velice kvalitní uživatelská podpora, jednak formou přehledně zpracované dokumentace [15, 16], jednak formou sociálních sítí na internetu. Například do skupiny na serveru Yahoo [17] se od roku 2000 připojilo již více než 40000 přispěvatelů (k datu 12.12.2009). Nesmíme

ani opomenout knihu Learning OpenCV autorského dua Gary Bradski a Adrian Kaehler [12], která problematiku počítačového vidění a OpenCV zpracovává velmi podrobně.

Z výše uvedených důvodů byla vybrána knihovna OpenCV jako nejvhodnější ke zpracování konečné aplikace. Komerční alternativou by mohla být knihovna IPP (Intel Performance Primitives), rovněž od firmy Intel [18, 19]. IPP lze využít samostatně, nebo společně s OpenCV, kdy dochází ke zlepšení výkonu kódu zapsaného v OpenCV. Podrobněji je tato problematika diskutována v kapitole 1 knihy [12]. Alternativou na bázi svobodného software může být knihovna Framewave [20], kterou podporuje společnost AMD.

3.2 Měření času

V oblasti zpracování obrazu často platí, že správný výsledek operace závisí nejen na přesnosti a korektnosti výpočtu, ale i na čase, ve kterém algoritmus k výsledku dojde. Obzvláště v aplikacích reálného času je běžné, že nejvhodnějším algoritmem se stává ten méně přesný avšak časově méně náročný oproti algoritmům s precizními výsledky, ale neúměrnými časovými nároky. Z tohoto důvodu byla při zpracování detekčního algoritmu věnována pozornost i časovým nárokům jednotlivých operací. Snahou bylo dosáhnout časově co nejefektivnějších výsledků. Časová náročnost operací zpracování obrazu zásadním způsobem narůstá s rozlišením zpracovávaných obrazů. Pokud se k řešení problému od počátku přistupuje se snahou o maximálně efektivní implementaci, lze vyvinout takový algoritmus, který může být v budoucnu použit na obrazových datech s vyšším rozlišením. Při vývoji algoritmu byly využity video sekvence v rozlišení HD, tedy 720 x 1280 pixelů.

V jazyku C/C++ existuje hned několik možností měření času. Několik z nich je přehledně popsáno v [21]. Následuje jejich stručná diskuze.

3.2.1. Knihovna C Time

Standardní knihovna v jazyce C++ obsahující funkce pro získání a manipulaci s časovými údaji. Specifikaci, lze nalézt na [22]. Lze pracovat s časem odvozeným z kalendářních údajů, případně s časem odvozeným od okamžiku spuštění programu. V obou případech je nejnižší jednotkou měření sekunda.

Při zpracování obrazových dat v reálném čase se předpokládá měření časových údajů řádově nižších a proto je tato knihovna pro tyto účely nevhodná.

3.2.2. Nativní měření času

Využívá knihoven <*sys/time.h*>, <*sys/types.h*> případně <*sys/resource.h*>, jejichž popis je k dispozici na [23]. Tyto knihovny jsou standardně obsaženy v systému Windows, nikoliv však na jiných platformách (nepřenositelnost kódu). Pro jejich využití nejsou vyžadovány žádné další externí knihovny, implementace je jednoduchá, ale opět poskytuje rozlišení naměřených hodnot pouze v jednotkách sekund.

3.2.3. Knihovna OpenMP

Tato sestava knihoven a direktiv pro překladač představuje multi-platformní podporu pro programování paralelních aplikací se sdílenou pamětí [24]. Tento projekt lze připojit k programům napsaným v jazyce C/C++ nebo Fortran. Funkce pro měření času lze využívat velmi jednoduše, podporuje měření ve více-vláknových aplikacích. Za nevýhody lze označit opět měření pouze s přesností na sekundy a dále nutnost použít speciálního překladače, který podporuje OpenMP[21].

3.2.4. Knihovny Boost

Projekt Boost je open-source soubor mnoha velmi užitečných a často využívaných knihoven a funkcí pro jazyk C++ [25]. V rámci tohoto projektu udržuje Beman Dawes knihovnu pro měření času [26]. Implementace je velmi jednoduchá (příklad v pseudokódu 3.1), přenositelná mezi platformami, poskytuje výsledky s rozlišením v mikrosekundách ačkoliv přesnost je poněkud menší. Za nevýhodné lze označit nutnost připojení externí knihovny k projektu.

Výpis pseudokódu 3.1: Využití knihovny Boost k měření času.

```
#include <boost/timer.hpp>

...

// Spuštění měření času
boost::timer t;

...

// Úsek kódu určený pro měření

...

// Zastavení měření a uložení výsledků do proměnné
double time = t.elapsed();


...
```

3.2.5. Knihovna WinPAPI

Projekt PAPI (Performance Application Programming Interface) je uceleným souborem softwarových entit, které umožňují programátorům zpřístupnit hardwarové čítače moderních mikroprocesorů [27]. Tyto čítače registrují události a signály spojené se stavem mikroprocesoru. Využitím knihovny WinPAPI lze získat informace o

- časování spuštěných procesů,
- stavu pamětí v procesoru,
- kolizích při přístupu ke sdíleným prostředkům,
- pipeliningu (paralelním zpracování instrukcí),
- informace o vláknech v případě více vláknových aplikacích, atd.

Knihovna utváří tzv. middleware vrstvu mezi programátorem a hardwarem mikroprocesoru. V rámci této vrstvy lze rozeznat tři úrovně, které se zavádí různou úroveň abstrakce [27]:

- Entity spodní vrstvy přímo obsluhují hardwarové události. Práce na této vrstvě klade větší nároky na programátora, nevyžaduje takovou režii jako vyšší vrstvy a proto dává nejpřesnější výsledky měření. Programátor má pod kontrolou všechny operace.
- Entity horní vrstvy umožňují jednoduchým způsobem spouštět, zastavovat a vyčítat naměřené základní údaje. Nepokrývají plnou škálu možností entit z nižší vrstvy, ale práce s nimi je velmi jednoduchá i pro nezkušeného programátora.

- Grafické prostředí pro koncového uživatele, nástroje pro vizualizaci statistik měření.

Knihovna PAPI je při měření času schopná generovat výsledky s přesností v řádech mikrosekund, na úrovni horní vrstvy ji lze velmi snadno implementovat, nevýhodou je nutnost instalace knihovny v systému a nepřenositelnost kódu na jiné platformy. Z uvažovaných alternativ poskytuje nejvhodnější přesnost měření a proto je v projektu implementována právě tato knihovna. Aktuální verzi lze získat na stránkách [27], kde je k dispozici i veškerá specifikace.

3.3 Struktura projektu

Zápis algoritmu je členěn do několika modulů, které sdružují související operace. Toto dělení je užitečné hned z několika důvodů. Na počátku složitá úloha se dělí na dílčí jednodušší úlohy, které jsou snáze řešitelné. Při vhodném návrhu lze každý z modulů samostatně udržovat a při zachování vstupně výstupních parametrů může na každém modulu pracovat jiný tým programátorů.

Tam kde to bylo vhodné byl použit objektový přístup, který umožňuje celý modul „zabalit“ do objektů. Objekty poté zpřístupňují svému okolí – tedy ostatním modulům – pouze vybrané funkce. Jiné funkce jsou naopak před okolím skryty. Tento přístup je výhodný u rozsáhlých programů, kde pomáhá udržet přehlednost a čitelnost zdrojových kódů a zabraňuje případným nekonzistencím v programu. Více o principech objektového programování najde čtenář například v publikaci [28].

Následuje přehled a stručná charakteristika jednotlivých modulů:

- Modul ***Calibration*** udržuje relaci mezi vzdáleností ve scéně s jednotkou v centimetrech a vzdáleností v obraze s jednotkou v pixelech. Ke správné funkci programu je vyžadován korektní průběh kalibrace.
- Modul ***Stopwatch*** tvoří pouze obálku k implementaci knihovny pro měření času WinPAPI.
- Modul ***Parametric Surface*** zajišťuje kompenzaci nerovnoměrného osvětlení ve scéně pomocí jednoho Bézierova bi-kubického plátu.
- Modul ***Frame Stamp*** je nejsložitějším modulem celé aplikace. Zajišťuje razítkování (klasifikaci) každého snímku vstupní video sekvence. Provádí kontrolu okrajů, hrubou a přesnou lokalizaci termostatu ve scéně a euklidovskou transformaci.

- Modul ***Mask*** definuje sledované vlastnosti termostatu a stará se o získání dat ze vstupního snímku, která jsou použita pro klasifikaci nalezeného objektu.
- Modul ***Text Output*** zajišťuje textové výpisy do konzolového okna aplikace. Sdružení textových výpisů z různých míst v programu do jednoho bloku umožňuje snadnou a přehlednou modifikaci jednotlivých zpráv, snadno lze implementovat například i různé jazykové varianty výpisů.

Každý modul je složen ze tří souborů:

- xxx.cpp – zdrojový kód implementovaných metod,
- xxx.h – hlavičkový soubor obsahující prototypy metod, definice výčtových typů, objektů a maker,
- xxx_setup.h – další hlavičkový soubor, který obsahuje především parametry daného modulu. Ve většině případů jde o definice pomocí makro instrukcí preprocesoru. Veškeré nastavení celého projektu lze provádět v souborech tohoto typu. Pro jeden zdrojový soubor typu *.cpp obvykle postačuje jeden hlavičkový soubor. Metodika použitá v tomto projektu zvyšuje přehlednost kódu a proto ji autor považuje za vhodnější. Pouze pro zajímavost uveďme, že celkový počet parametrů v tomto projektu je více než 120.

Obrázek 3.1 shrnuje souborovou strukturu celého projektu.

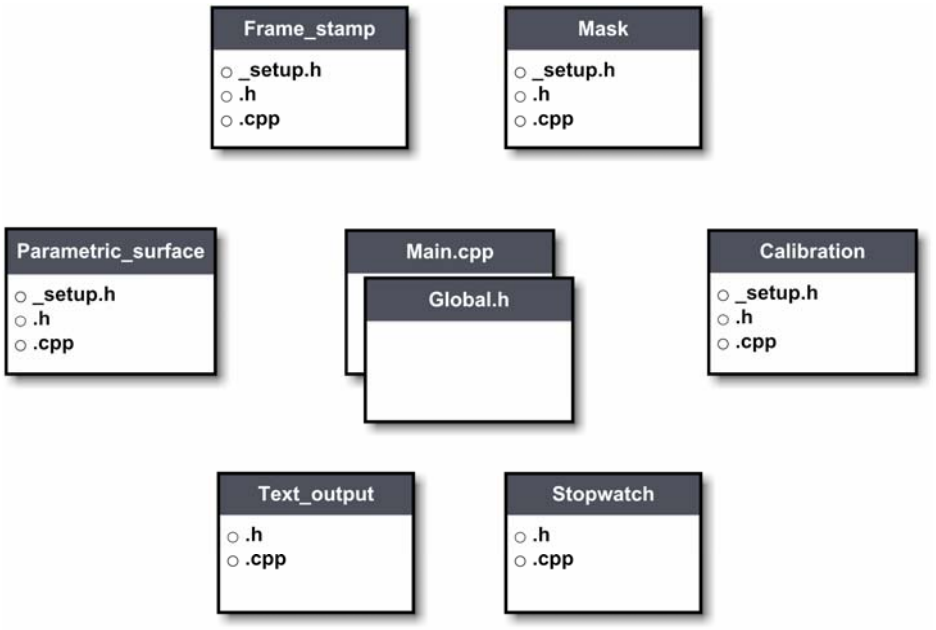
3.4 Popis algoritmu

Ze vstupní video sekvence jsou posupně odebírány jednotlivé snímky, ve kterých je kompenzováno nerovnoměrné osvětlení. Následně je snímek předán do modulu ***Frame Stamp***, který jej klasifikuje.

Idea algoritmu je taková, že ke změně ve scéně může dojít pouze vnějším zásahem obsluhy pracoviště. To znamená že operátor do snímané scény vloží nějaký objekt, případně jej vyjme. Rozpoznat okamžik, kdy operátor zasahuje do scény lze pomocí kontroly okrajů snímané oblasti – není nutné zpracovávat celý obraz. Tato operace je prvním krokem při klasifikaci. Snímky, u kterých je nalezen porušený okraj není nutné dále zpracovávat, protože v nich dochází k manipulaci s objekty. V opačném případě se pokračuje lokalizací objektů ve scéně.

V případě nalezení korektního objektu je navržena a provedena euklidovská transformace zpracovávaného obrazu s cílem rotovat a posunout vstupním obrazem takovým způsobem, aby byl objekt termostatu přesunut do definované pozice v obrazu. V tomto okamžiku končí klasifikace vstupního obrazu.

Na transformovaný obraz je aplikován modul ***Mask***, ve kterém jsou definovány klíčové oblasti zájmu na zkoumaném termostatu. Z definovaných oblastí jsou odebrána obrazová data, která jsou vyhodnocena a klasifikována v rámci množiny přípustných stavů.

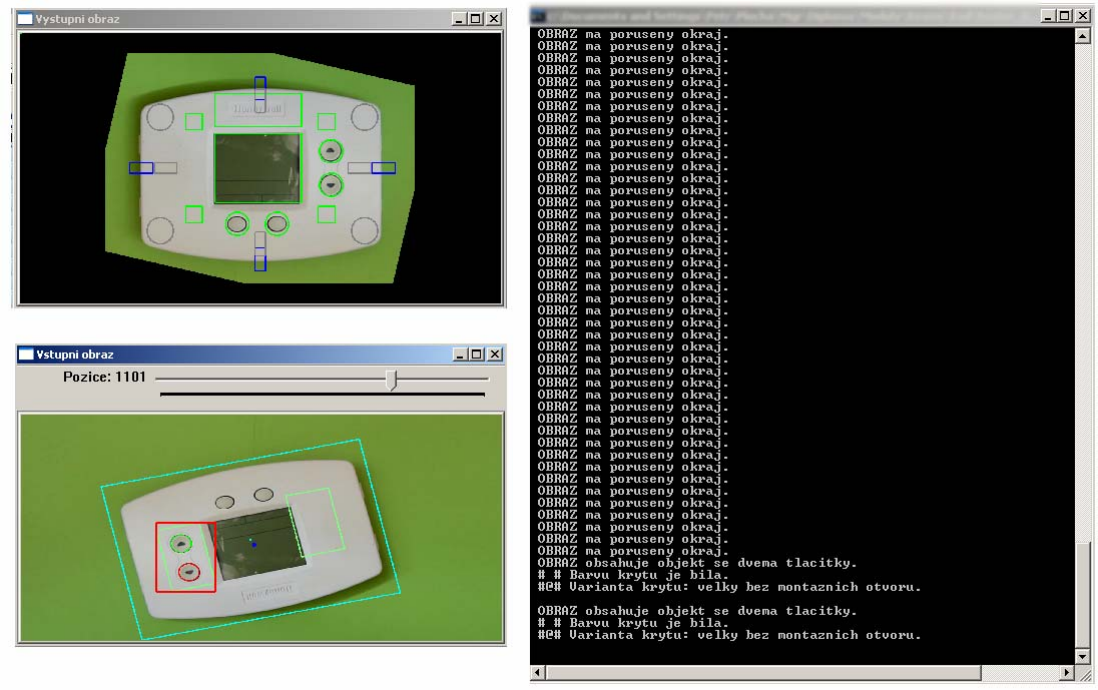


Obr. 3.1: Souborová struktura projektu.

3.5 Demonstrační video

K této práci je v elektronické formě přiložena mimo jiné i spustitelná verze programu (Projekt.exe) společně s demonstračním videem (P1000257.avi). Video i program musí být umístěny ve stejném adresáři. Po spuštění programu se otevřou tři okna, viz obr. 3.2. V pravé polovině obrázku je konzolové okno, kam se vypisují údaje o zpracovávané scéně. Především potom výsledky klasifikace jednotlivých snímků a popis nalezeného termostatu. V levém spodním okně je vidět vstupní obraz video sekvence, ve kterém jsou naznačeny oblasti, které algoritmus využil v průběhu lokalizace termostatu. V horní polovině obrazovky je uživatel informován o aktuální pozici ve video sekvenci společně s pořadovým číslem snímku. Levé horní okno ukazuje výstupní obraz po

přesunu nalezeného objektu do definované polohy. V tomto snímku jsou rovněž naznačeny jednotlivé oblasti masky, ve kterých se kontrolují požadované parametry. Vykonávání programu lze přerušit stiskem tlačítka „***p***“, jeho opětovným stlačením program pokračuje ve své činnosti. Ukončit aplikaci lze kdykoliv stiskem tlačítka „***ESC***“. Demonstrační video je sestavno tak, aby demonstrovalo všechny zásadní vlastnosti algoritmu. Po testu okrajových oblastí je ilustrován proces kalibrace následovaný viditelným vyvážením nerovnoměrného osvětlení a nakonec jsou do scény vkládány různé typy termostatu, současně jsou do konzolového okna vypisovány výsledky detekce.



Obr. 3.2: Demonstrační aplikace.

3.6 Modul Parametric Surface

Kapitola 2.2 představuje postup, kterým lze kompenzovat nerovnoměrné osvětlení ve snímané scéně. Modul Parametric Surface je implementací tohoto postupu. Ze dvou diskutovaných metod modelování plochy (Bézierova a Coonsova placha) byla pro realizaci vybrána první jmenovaná především z důvodu snazší algoritmizace. Pro účely tohoto projektu poskytne dostatečnou plasticitu jeden plát. Následující pseudokód naznačuje implementaci Bézierovy plochy v prostředí MATLAB.

3.6.1. Implementace v prostředí Matlab

Matice řídících bodů *Pz* je naplněna hodnotami jasu ze vstupního obrazu. Pseudokód vybírá hodnotu ze vstupního obrazu bodově, v praktické aplikaci je výhodnější uvažovat průměrnou hodnotu z určitého okolí, nebo řídící hodnoty váhovat jiným způsobem (mediánový filtr, ...). Matice 4 x 4 řídících bodů je nutné normalizovat do rozsahu $\langle 0,1 \rangle$. Normalizace jasových hodnot předpokládá 8mi bitovou reprezentaci.

Výpis pseudokódu 3.2: Řídící body.

```
% rovnomerne rozlozeni ridicich bodu po plose celeho obrazu
% x-ove souradnice
Px(:,:) = [meshgrid(1:floor((xmax-1)/3):xmax)]
% y-ove souradnice
Py(:,:) = [meshgrid(1:floor((ymax-1)/3):ymax)']

% jasove hodnoty ridicich bodu
Pz(:,:) = Vstupni_obraz(Px(1,:),Py(:,1))

% normalizace ridicich bodu
Px (:,:) = Px (:,:)/xmax
Py (:,:) = Py (:,:)/ymax
Pz (:,:) = Pz (:,:)/255
```

Inverzního charakteru kompenzační plochy vůči ploše původní je dosaženo překlopením řídících bodů kolem stejnosměrné složky. Definice bázové matice Bézierovy bikubické plochy vychází z Bernsteinových polynomů třetího řádu, tak jak je odvozeno v (2.13).

Výpis pseudokódu 3.3: Inverze řídících bodů, bázová matice Bézierovy plochy.

```
% stredni hodnota jasove slozky ridicich bodu
Offset = (mean(mean(Pz (:,:))))
% a jeji odstaneni
Pz = Offset - Pz


% bazova matice Bezierovy plochy
M = [-1  3 -3 1;...
      3 -6  3 0;...
     -3  3  0 0;...
      1  0  0 0]
```

Následuje jádro celého algoritmu. Výpočet kompenzační plochy dle (2.14) pro každý bod vstupního obrazu. V každé iteraci je nutné *u*, *v* normalizovat do rozsahu $\langle 0,1 \rangle$ (viz. 2.14, 2.15).

Výpis pseudokódu 3.4: Výpočet kompenzační plochy.

```
% zjisteni rozmeru obrazu
[xmax ymax] = size(Vstupni_obraz(:,:))


% konstrukce kompenzacni plochy
for i= 1:xmax
    for j = 1:ymax
        % normalizace souradnic u,v
        u = (i-1)/(xmax-1)
        v = (j-1)/(ymax-1)
        U = [u^3 u^2 u 1]
        V = [v^3 v^2 v 1]

        Kompenzacni_plocha(i,j) = U*M*Pz*M'*V'
    end
end
```

Vlastní kompenzace spočívá v součtu původní plochy s plochou kompenzační. Matice *Pz* byla před výpočtem normalizovala a proto i hodnoty *Kompenzacni_plocha* jsou pouze z rozsahu $\langle 0,1 \rangle$. Návrat do původního 8mi bitového rozlišení zajistí násobení konstantou ($2^8 - 1 = 255$).

Výpis pseudokódu 3.5: .

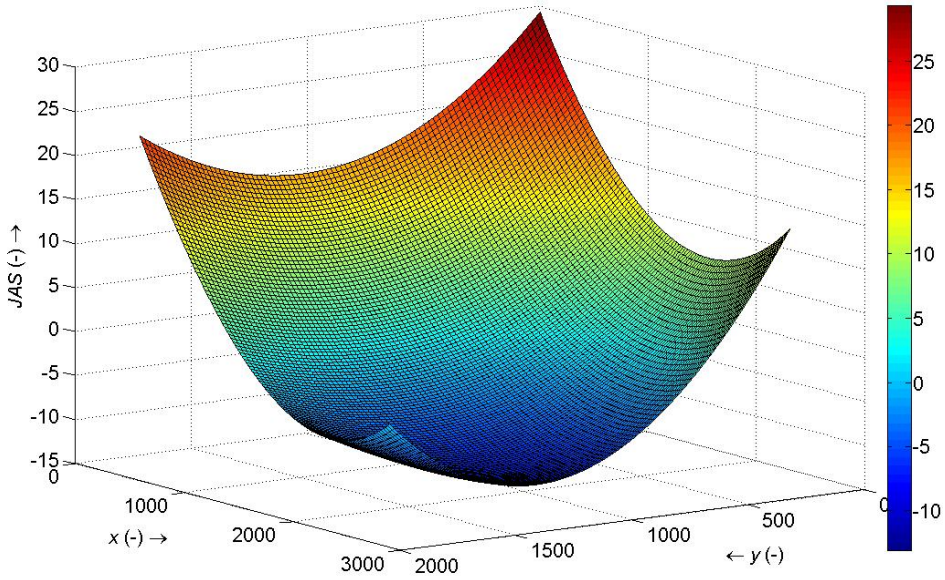
```
Vystupni_obraz = Vstupni_obraz + (255*Kompenzacni_plocha)
```

Na obrázku 3.3 je zobrazena kompenzační plocha pro snímek z obr. 2.3 až 2.5. Tam, kde původní plocha měla nízké hodnoty, dosahuje kompenzační plocha hodnot vysoce kladných. Naopak v místě původního maxima lze nalézt hodnoty záporné. Výsledná plocha po kompenzaci (obr. 3.4) je součtem ploch z obr. 3.3 a 2.5. Navzdory vysoké úrovni šumu je dobře patrná rovnoměrnější distribuce jasu v celé scéně při zachování přibližně stejné průměrné hodnoty. Zlepšení je patrné i v ekvalizované verzi obrazu (více homogenní nežli protějšek na obr.2.3 vpravo). Ekvivalent k histogramu z obr. 2.4 je uveden na obr. 3.6. Zlepšení jasových charakteristik testovacího snímku po

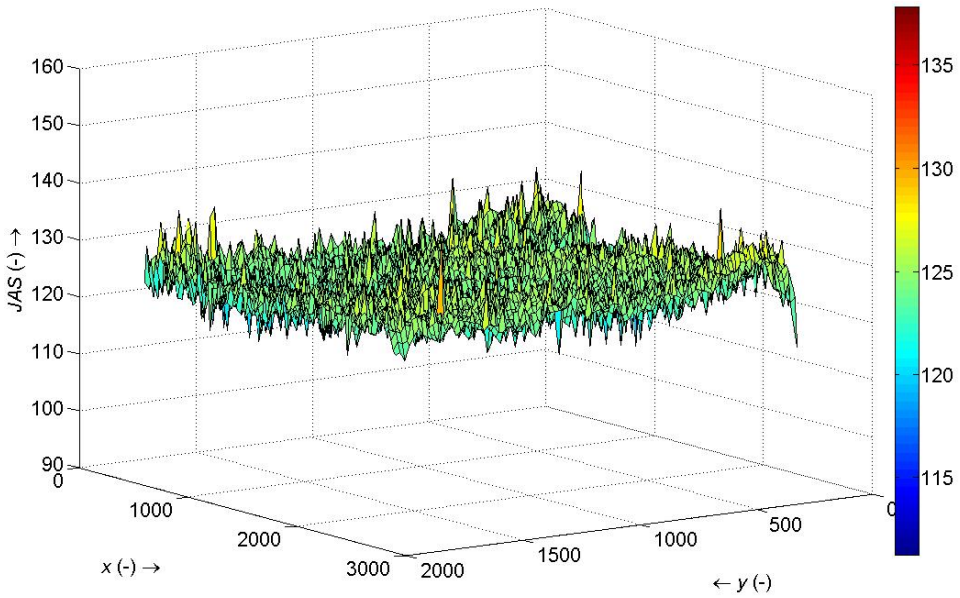
kompenzaci shrnuje tab. 3.1, která porovnává statistické vlastnosti původního obrazu a jeho vylepšeného protějšku.

Tab. 3.1: Statistické vlastnosti obrazů z obr. 2.3 až 2.5 a obr. 2.7 až 2.9

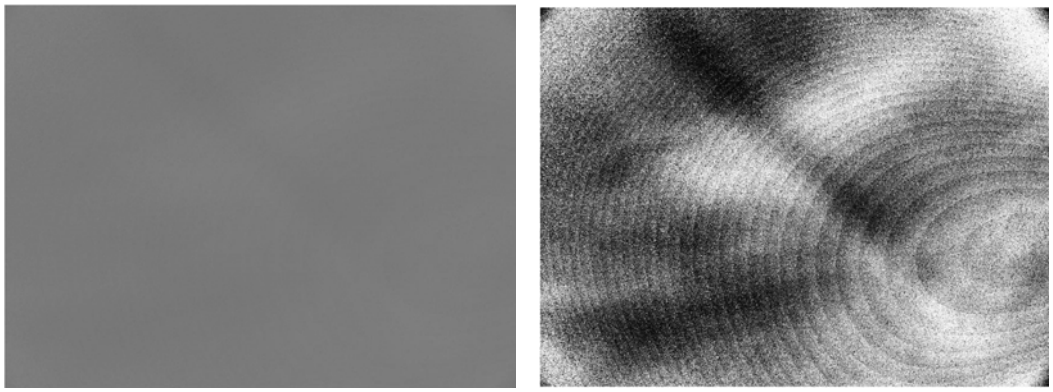
	Původní obraz	**Výsledný obraz**
Střední hodnota	122.31	122.32
Standardní odchylka	9.77	2.55



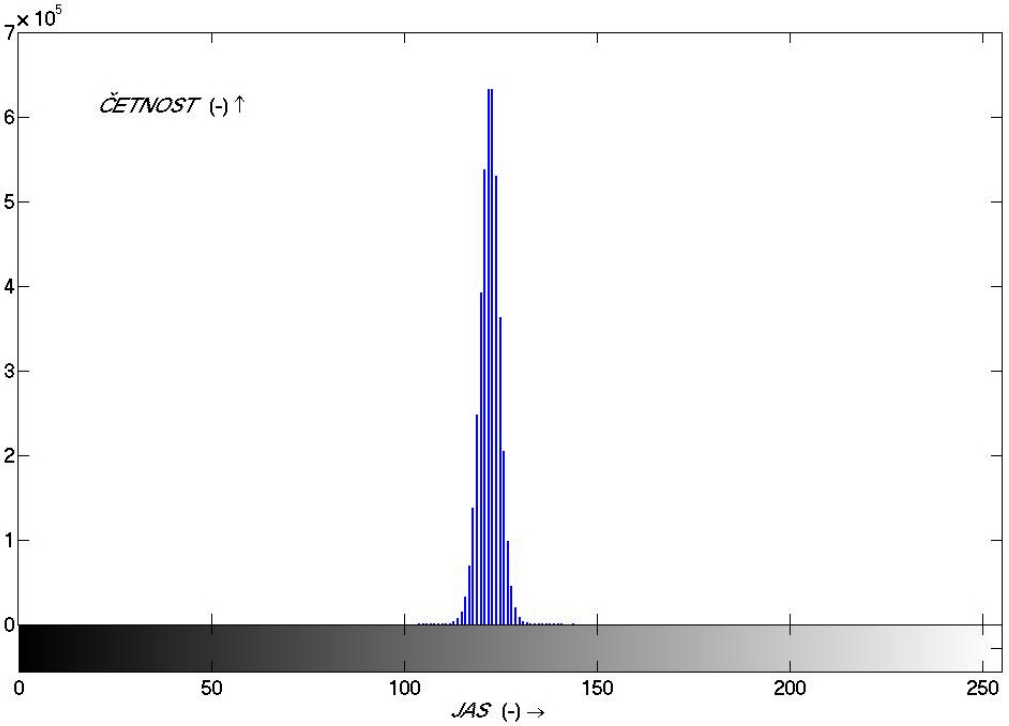
Obr. 3.3: Kompenzační plocha obrazu 2.3 až 2.5.



Obr. 3.4: Plocha z obr. 2.5 po kompenzaci.



Obr. 3.5: Vlevo obraz 2.3 po kompenzaci, vpravo navíc ekvalizováno.



Obr. 3.6: Histogram obrazu z obr. 2.7, 2.8

3.6.2. Implementace v prostředí C++

Implementace výše popisovaného nástroje s sebou nese v jazyku C++ určitá specifika. Jak je patrné z obr. 3.3 kompenzační plocha nabývá kladných i záporných hodnot. Tato skutečnost neznamená žádnou komplikaci v prostředí MATLAB, ale v knihovně OpenCV je se zápornými hodnotami problém. Řešením této situace je rozklad kompenzační plochy na dvě dílčí, kde první z nich obsahuje pouze kladné hodnoty a druhá pouze absolutní hodnotu záporných částí původní kompenzační plochy. Samotná

kompenzace se potom děje tak, že kladná část plochy se přičítá k originálnímu obrazu a záporná část plochy v absolutní hodnotě se naopak odečítá.

Aby nedocházelo k barevným posunům je návrh plochy proveden v barevném modelu HLS, konkrétně nad jeho jasovou složkou L.

Díky optimalizačním krokům trvá korekce osvětlení pouze cca 14 milisekund pro obrazy o HD rozlišení na PC s konfigurací Intel Core 2 Duo, 1,66GHz, 1GB RAM.

3.7 Modul Frame Stamp

Tento modul mimo jiné zajišťuje lokalizaci termostatu. Z důvodu snížení časové náročnosti je tento proces rozdělen do dvou částí. Nejprve probíhá tzv. hrubá lokalizace, jejímž cílem je určit přibližnou oblast, kde se vyskytuje termostat. V dalším kroku potom algoritmus hladké lokalizace nemusí pracovat nad celým obrazem, ale pouze nad omezeným výřezem. Hrubá lokalizace využívá metody semínkového zaplavování nad podvzorkovaným vstupním obrazem.

Pro účely hladké lokalizace je nutná znalost nějakého orientačního prvku na krytu termostatu (obr. 1.1). Jako nejvhodnější se jeví tlačítka v pravé části zařízení. Jejich barva je na všech variantách krytů konstantní, mají unikátní tvar v rámci celého termostatu. Materiál tlačítek není natolik náchylný k odleskům jako materiál displeje. K detekci kruhových obrysů tlačítek lze využít Houghovu transformaci kružnic, která byla diskutována v rámci kapitoly 2.3.

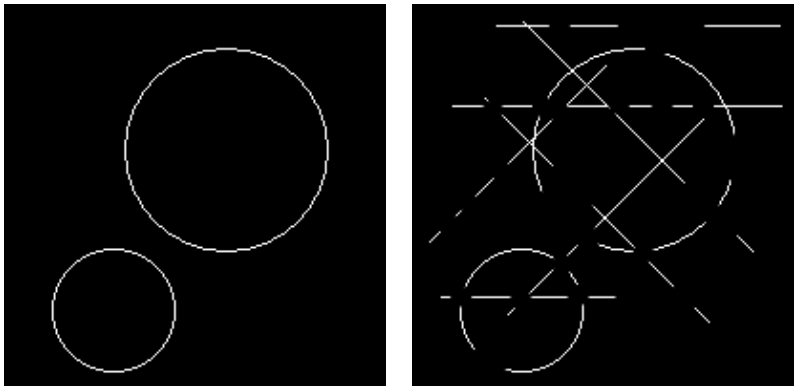
3.7.1. Implementace Houghovy transformace kružnic

Pro demonstrační účely jsme sestavili dva testovací snímky (viz. obr. 3.7), každý o rozměru 200 x 200 (px). Vlevo je zobrazen dokonalý výstup z hranového detektoru. Původní obraz obsahoval dva kulaté objekty o poloměrech r_1 = 32 px, r_2 = 53 px. Vpravo je potom umístěn výstup nedokonalé hranové detekce totožného obrazu. Vidíme zde několik výpadků hran na místech, kde bychom hrany očekávali, objevují se i hrany falešné.

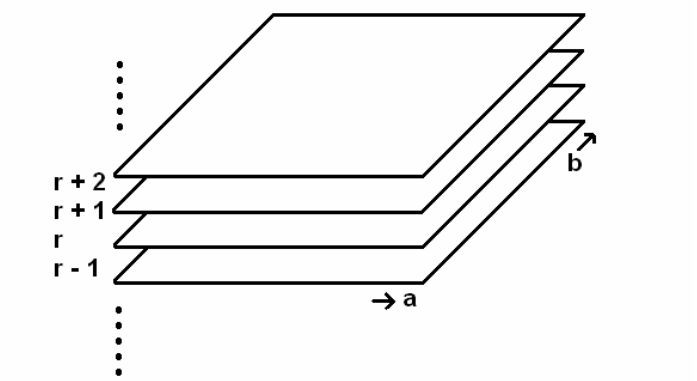
Obrázek 3.8 naznačuje implementaci parametrického prostoru jako třírozměrného akumulátoru. Každá vrstva představuje řez parametrickým prostorem pro konst. *r*.

Obrázky 3.9 a 3.10 přinášejí výsledky HT pro testovací obrazy. Na obr. 3.9 nahoře můžeme vidět 32 vrstvu akumulátoru při optimálním vstupním obrazu. Je zde patrné

výrazně ostré maximum, které svou polohou determinuje menší z kružnic na testovacím obrazci. Dole potom vidíme výsledky pro porušenou variantu testovacího obrazce. Zřetelný je nárůst akumulovaných hodnot po celém řezu, rovněž si povšimněme, že špičková hodnota maxima se snížila oproti předchozí variantě.

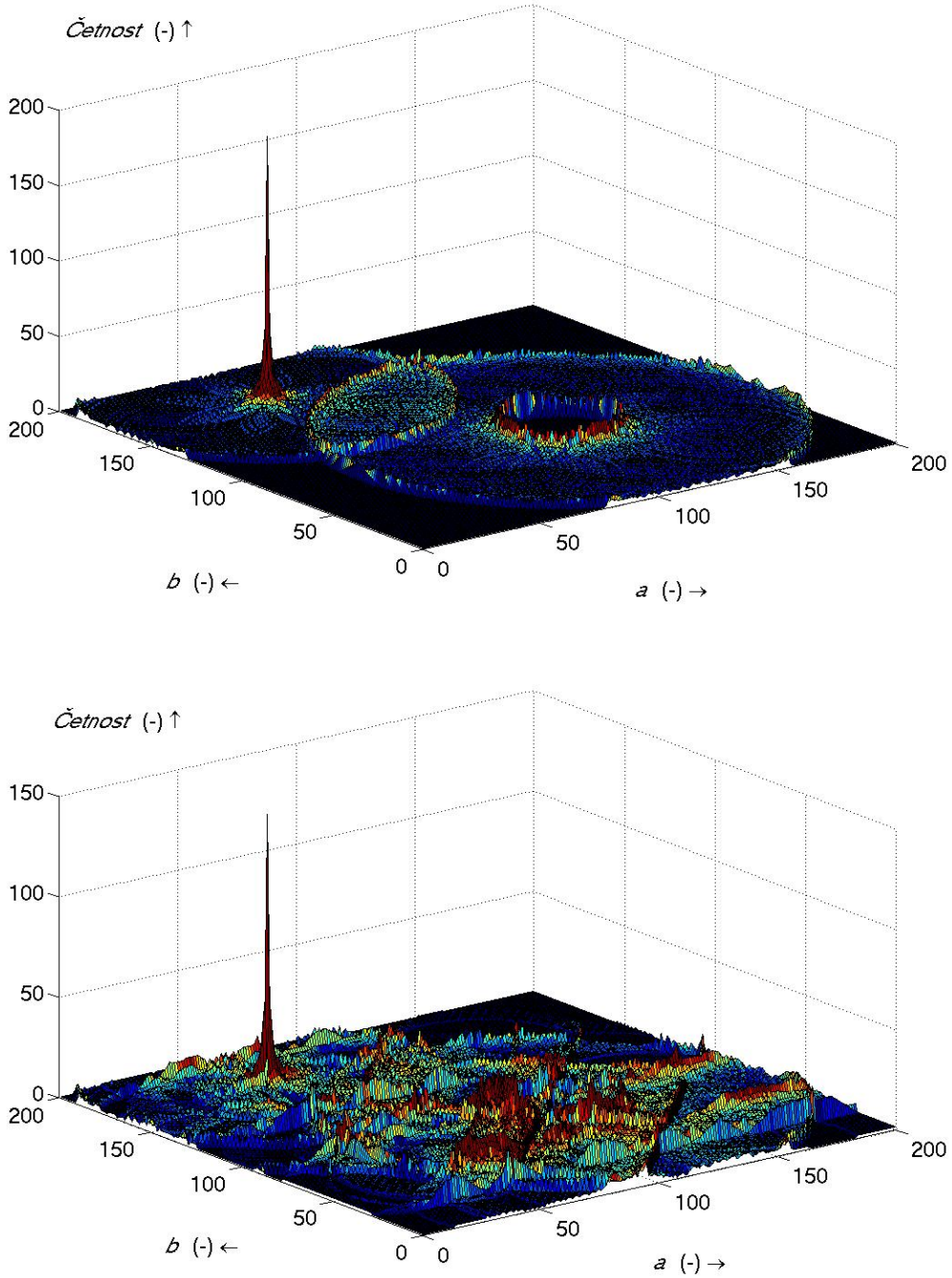


Obr. 3.7: Testovací obrazy pro HT.

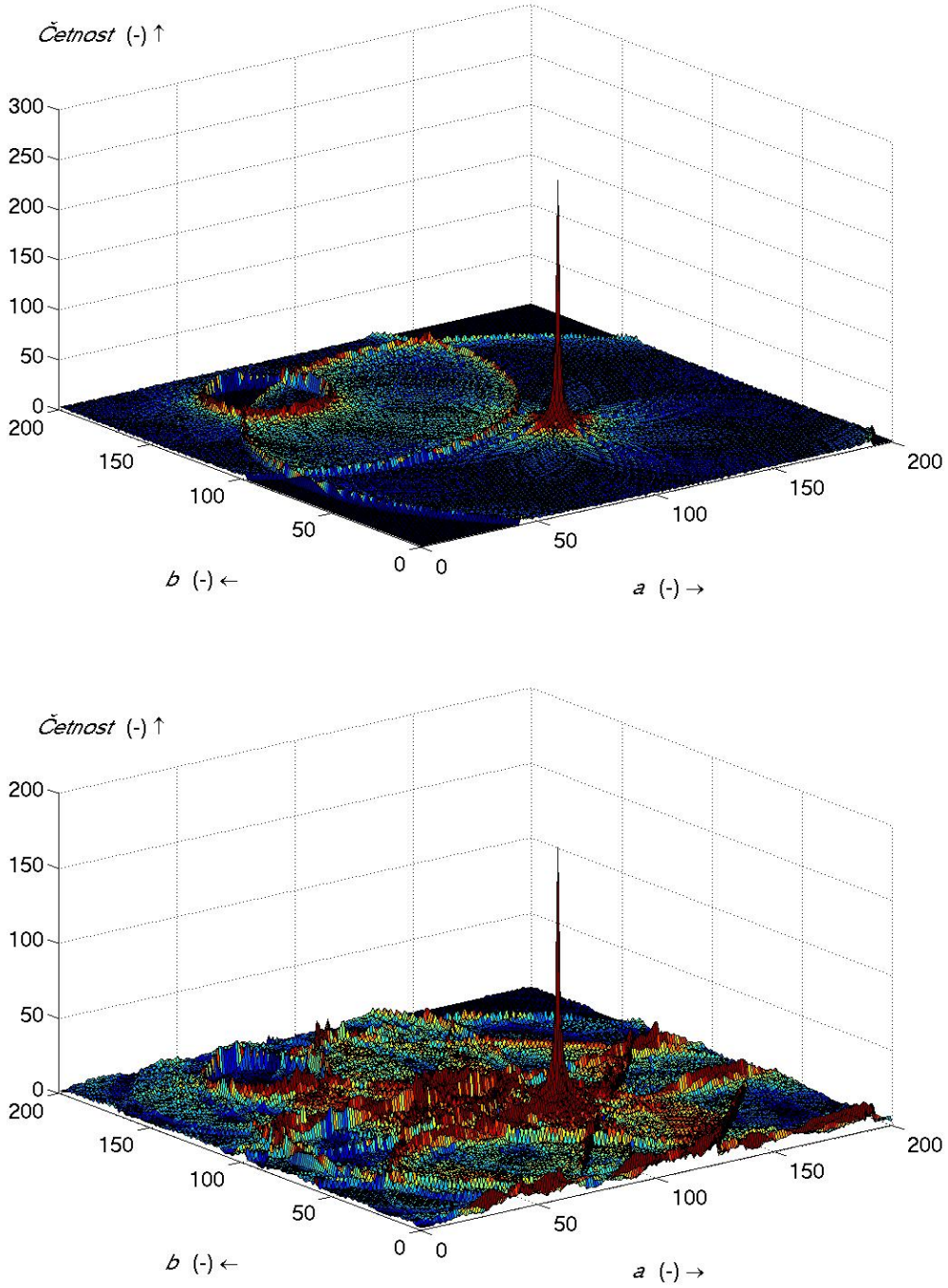


Obr. 3.8: Parametrický prostor jako třírozměrný akumulátor.

Na obr. 3.10 vidíme 53 vrstvu akumulátoru, zde očekáváme maximum pro větší z dvojice kružnic na testovacím obrazci. Platí zde shodné závěry jako u předchozího popisu. Na obou obrázcích si rovněž povšimněme, že kružnice odlišného poloměru jsou na daném řezu téměř dokonale atenuovány.



Obr. 3.9: Akumulátor pro kružnici o poloměru r_1=32 px,
nahoře pro ideální vstupní obraz, dole pro obraz porušený.



Obr. 3.10: Akumulátor pro kružnici o poloměru r_2=53px,
nahoře pro ideální vstupní obraz, dole pro obraz porušený.

Díky optimalizačním krokům se podařilo dosáhnout, že proces známkování trvá přibližně 100 milisekund. Velký podíl na tom má euklidovská transformace, kterou provádíme v případě, že nalezneme v obraze termostat. Ukazuje se, že Houghova

transformace, pokud je správně parametrizována, může pracovat ve velmi efektivních časech se spolehlivými výsledky.

3.8 Modul Mask

Pro rozpoznání velikosti krytu se využívá metod prahování, přítomnost montážních otvorů vyhodnocuje algoritmus na základě statistických vlastností obrazových dat v příslušných obrazech. Pro klasifikaci barvy krytu se využívá euklidovské vzdálenosti naměřené barvy od barevných předloh. Všechny výše uvedené parametry jsou rozpoznávány s velmi vysokou jistotou.

Nevyřešeným problémem zůstává zpracování a rozpoznání loga. Na základě testování v prostředí MATLAB, lze očekávat, že implementace prostého konvolučního porovnávání databáze s naměřenými daty bude poskytovat dostatečně spolehlivé a selektivní výsledky.

Časové nároky tohoto modulu jsou řádově v jednotkách milisekund. Pracuje se výhradně s malými bloky obrazových dat, proto v tomto modulu mohou být využity sofistikovanější metody.

4 Závěr

Teoretická část této práce přináší především vlastní autorův návrh techniky pro kompenzaci nerovnoměrného osvětlení ve snímané scéně. Navržená technika je následně implementována do komplexního algoritmu, který zabezpečuje spolehlivou detekci prvků termostatu. Kompenzace osvětlení se ukazuje jako velmi důležitou součástí celého algoritmu především při určování barvy krytu termostatu. Díky této metodě bylo dosaženo výrazného zlepšení při klasifikaci získané barevné informace z reálného obrazu. Nezanedbatelný přínos navržené metody lze shledat i při segmentaci obrazových dat.

V průběhu realizace algoritmu se některé knihovní funkce OpenCV ukázaly jako ne příliš efektivní, proto byly nahrazovány pointerovou aritmetikou. Díky tomuto přístupu čas potřebný pro vyhodnocení snímku v rozlišení HD (720 x 1280 pixelů) nepřekračuje 200 milisekund.

Klíčovými prvky sestaveného algoritmu jsou Houghova transformace a algoritmus semínkového zaplavování. Společně tyto dvě techniky umožňují v efektním čase identifikovat objekt termostatu ve scéně. Algoritmus je dále vybaven vlastním kalibračním mechanismem, který z explicitní znalosti rozměrů kalibračního obrazce dokáže odvodit vztah mezi rozměrem v obraze a ve scéně.

Z požadovaných vlastností se nepodařilo z časových důvodů vyřešit rozpoznávání zákaznických log. V textu je ovšem krátce nastíněn způsob možného řešení této problematiky. Ostatní parametry krytu jsou algoritmem vyhodnocovány s přesností blízké 100%.

Zdrojové kódy jsou systematicky členěny do nezávislých objektů. Tento způsob práce se zdrojovým kódem usnadňuje jeho čitelnost a budoucí úpravy tak činí méně náročnými. Jako možné rozšíření se nabízí například využití metod pro adaptivní nastavování parametrů algoritmu, kterých je více než 120. Rozšíření databáze termostatů o nové varianty není problematické, díky způsobu zápisu zdrojových kódů. Snadno z jednoho místa v kódu lze rovněž změnit jazyk textových výpisů do konzolového okna.

Literatura

[1] ŠONKA, Milan , HLAVÁČ , Václav. *Počítačové vidění*. 1992. vyd. Praha : Grada, 1990. 252 s. ISBN 80-85424-67-3.

[2] GONZALES, Rafael C., WOODS, Richard E. *Digital Image Processing*. 2nd edition. New Jersey : Prentice Hall, 2002. 793 s. ISBN 0-201-18075-8.

[3] RUSS, John C. *Image Processing Handbook*. 4th edition. Boca Raton : CRC Press, 2002. 723 s. ISBN 0-8493-1142-X.

[4] BAXANT, Petr. *Světelná technika*. Brno : [s.n.], 2006. 82 s. Elektronické skriptum.

[5] ŽÁRA, Jiří, et al. *Moderní počítačová grafika*. 1. vyd. Brno : Computer Press, 2004. 609 s. ISBN 80-251-0454-0.

[6] *Leptonica* [online]. 2009, Jan 11, 2009 [cit. 2010-05-20]. Seed Filling and Connected Components. Dostupné z WWW: <http://www.leptonica.com/filling.html>.

[7] VANDEVENNE, Lode . *Lode's Computer Graphics Tutorial* [online]. 2004, 2004 [cit. 2010-05-20]. Flood Fill. Dostupné z WWW: <http://www.student.kuleuven.be/~m0216922/CG/floodfill.html>.

[8] BENEŠ, Radek. *Využití metod zpracování signálů pro zvýšení bezpečnosti automobilové dopravy*. Brno, 2009. 71 s. Diplomová práce. Vysoké učení technické v Brně.

[9] SHAW, John R. . *CodeProject* [online]. 12 Mar 2004. 2 Feb 2004, 12 Mar 2004 [cit. 2010-05-20]. QuickFill: An efficient flood fill algorithm. Dostupné z WWW: <http://www.codeproject.com/kb/gdi/QuickFill.aspx>.

[10] *Open Computer Vision Library* [online]. c2009 [cit. 2009-12-10]. Text v angličtině. Dostupný z WWW: <http://sourceforge.net/projects/opencvlibrary/>.

[11] Intel Corporation. *OpenCV* [online]. 2.0 / 2009-09-30. 2009 , last modified on 7 December 2009 [cit. 2009-12-10]. Text v angličtině. Dostupný z WWW: <http://en.wikipedia.org/wiki/OpenCV>.

[12] BRADSKI, Gary , KAEHLER, Adrian . *Learning OpenCV*. Mike Loukides; Robert Romano. Sebastopol : O'Reilly Media, Inc., c2008. 555 s. ISBN 978-0-596-51613-0.

[13] *Open Source Initiative OSI - The BSD License : Licensing The BSD License* [online]. Opensource.org, [2009] [cit. 2009-12-12]. Dostupný z WWW: <http://opensource.org/licenses/bsd-license.php>.

[14] AUJEZDSKÝ, Josef. *Právní aspekty volně šiřitelných počítačových programů* [online]. c1998-2009 [cit. 2009-12-08]. Dostupný z WWW: <http://www.root.cz/specialy/licence/>.

[15] *OpenCV 2.0 C Reference* [online]. c2009 [cit. 2009-12-08]. Dostupný z WWW: <http://opencv.willowgarage.com/documentation/index.html>.

[16] *Open Source Computer Vision Library : Reference Manual* [online]. Dec 8, 2000. U.S.A : Intel Corporation, 2010 [cit. 2010-05-20]. Dostupné z WWW: <http://www.cs.unc.edu/Research/stc/FAQs/OpenCV/OpenCVReferenceManual.pdf>.

[17] *Open Source Computer Vision Library Comm* [online]. c2009 [cit. 2009-12-10]. Dostupný z WWW: <http://tech.groups.yahoo.com/group/OpenCV/>.

[18] *Intel® Integrated Performance Primitives (Intel® IPP) 6.1* [online]. Intel Corporation, [2009] [cit. 2009-12-10]. Dostupný z WWW: <http://software.intel.com/en-us/intel-ipp/>.

[19] *Intel® Integrated Performance Primitives for Intel® Architecture : Reference Manual* [online]. Volume 2. Intel Corporation, 2008 [cit. 2009-12-12]. Dostupný z WWW: <http://www.intel.com/cd/software/products/asmo-na/eng/405959.htm>.

[20] *About Framewave* [online]. The Framewave Group., c2008 [cit. 2009-12-10]. Dostupný z WWW: <http://framewave.sourceforge.net/>.

[21] IGLBERGER, K. ; HEUBECK, B. ; JANDL, C. . *Time Measurement in C/C++* [online]. [s.l.] : University Erlangen-Nuremberg, December 15th-16th 2008 [cit. 2010-05-20]. Dostupné z WWW: <http://www10.informatik.uni-erlangen.de/Teaching/Courses/WS2008/SiWiR/Time.pdf>.

[22] *C++ Reference* [online]. 2009 [cit. 2010-05-20]. C Time Library. Dostupné z WWW: <http://www.cplusplus.com/reference/clibrary/ctime/>.

[23] *The Open Group Base Specifications* [online]. 2004 Edition. 2004 [cit. 2010-05-20]. <sys/time.h>. Dostupné z WWW: <http://www.opengroup.org/onlinepubs/000095399/basedefs/sys/time.h.html>.

[24] *The OpenMP API specification for parallel programming* [online]. May 5, 2010 [cit. 2010-05-20]. Resources. Dostupné z WWW: <http://openmp.org/wp/resources/>.

[25] *Boost C++ Libraries* [online]. 2010-05-27 [cit. 2010-05-20]. Boost Library Documentation. Dostupné z WWW: <http://www.boost.org/doc/libs/>.

[26] *Boost C++ Libraries* [online]. 1999, November 07, 2007 [cit. 2010-05-20]. Timer Library. Dostupné z WWW: <http://www.boost.org/doc/libs/1_43_0/libs/timer/timer.htm>.

[27] *PAPI* [online]. May 20 2010 [cit. 2010-05-20]. Performance Application Programming Interface. Dostupné z WWW: <http://icl.cs.utk.edu/papi/>.

[28] PRATA, Stephen. *Mistrovství v C++*. Praha : Computer Press, 2007. 957 s. ISBN 80-7226-339-0.