VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE VÝROBKŮ NA PÁSOVÉM DOPRAVNÍKU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. ALEŠ LÁNÍK
AUTHOR

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE VÝROBKŮ NA PÁSOVÉM DOPRAVNÍKU

DETECTION OF OBJECTS ON BELT CONVEYER

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE BC. ALEŠ LÁNÍK
AUTHOR

VEDOUCÍ PRÁCE ING. MICHAL ŠPANĚL
SUPERVISOR

BRNO 2008

Abstrakt

Tato diplomová práce se zabývá problematikou detekce objektů v obraze a jeho trasováním v čase. Nejprve je popsán teoretický základ pro předzpracování obrazu, filtraci obrazu, extrakci popředí a různé další obrazové příznaky. Dále je zpracován návrh a implementace detektoru. Tato část je konkrétně zaměřena na detekci objektů na dopravníkovém páse. Poslední část diplomové práce obsahuje shrnutí, závěr a dodatky.

Klíčová slova

Detekce objektů v obraze, barevné prostory, morfologické operace, detekce rohů a hran, extrakce popředí, segmentace obrazu, texturní příznaky.

Abstract

In this master thesis, object's detection in image and tracking these objects in temporal area will be presented. First, theoretical background of the image's preprocessing, image filtration, the foreground extraction, and many others various image's features will be described.

Next, design and implementation of detector will be processed. This part of my master thesis containes mainly information about detection of objects on belt conveyer

Finally,results, conclusion and many supplementary data such as a photography camera's location will be shown.

Keywords

Detection objects in image, color spaces, morphological operations, corner and boundary detection, foreground extraction, image segmentation, texture features.

Citace

Láník Aleš : Detekce výrobků na dopravníkovém pásu, diplomová práce, Brno, FIT VUT v Brně. 2008

Detekce výrobků na pásovém dopravníku

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením ing. Michala Španěla. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum

Poděkování

Rád bych na těchto pár řádcích poděkoval mému vedoucímu diplomové práce Ing. Michalu Španělovi za odborné vedení mé diplomové práce a také za nesčetné konzultace. Dále bych poděkoval firmě Techservis Láník Boskovice za poskytnutí testovacích dat a za vřelý přístup.

Obsah

1 Úvod

S rozvojem výpočetních technologií, především ve výpočetním výkonu, vzrůstající kapacitě pamětí a radikálním snižováním pořizovacích nákladů hardwaru, vyvstává čím dál větší potřeba uplatňovat tyto technologie v průmyslu, ve kterém nepopiratelně přináší kvalitnější a levnější výrobu. Dále pak výpočetní technologie pomáhají v kvalitnějším a inteligentnějším plánování a v podpoře rozhodování.

Lze dokonce říci, že pořizování informačních technologií a jejich začlenění do všech sfér podnikové hierarchie, je v dnešní době nezbytnou záležitostí, na níž může záviset celková konkurenceschopnost podniku na trhu.

Tato diplomová práce, si klade za cíl, navrhnout a implementovat systém na sledování a počítání objektů na jedoucím dopravníkovém páse. V první části (tedy v semestrálním projektu) jsem se zabýval výhradně teoretickým návrhem a rozborem metod, které by bylo možno využít při konstrukci cílového systému. Dále byly během semestrálního projektu pořízeny testovací data.

V navazující diplomové práci jsem provedl implementaci systému, jeho skutečné hardwarové provedení a začlenění do výrobní linky ve firmě Techservis Láník Boskovice.

V dalších odstavcích shrnu stručný obsah jednotlivých kapitol této práce a jejich smysl pro implementaci systému.

Kapitola 2 se zabývá metodami pro zpracování obrazu. Jmenovitě kapitola obsahuje podkapitoly zabývající se barevnými prostory, ekvalizací histogramu, morfologickými operacemi, algoritmy pro filtraci obrazu, metodami pro extrakci popředí od pozadí, dále je tu podkapitola zabývající se algoritmy pro extrakci pomocných příznaků a poslední podkapitola rozebírá segmentaci obrazu.

Kapitola 3 se zabývá implementací celého systému. V první podkapitole 3.1 je navrženo schéma detektoru. Jednotlivé bloky tohoto schématu jsou dále rozebrány v dalších podkapitolách.

První podkapitola 3.2 se zabývá vstupem systému (senzorem) a fyzickou realizaci tohoto vstupu. Dále je v této podkapitole rozebráno předzpracování obrazu. Následující podkapitolou je detekce blobů a jejich trasování 3.3 , ve které se rozebírají konkrétní použité algoritmy segmentace obrazu na jednotlivé bloby, reprezentující detekované objekty, a jejich následné trasování v čase. V podkapitole 3.4 se zabývám řešením chybových stavů, zejména detekcí a ošetřením spojených blobů. Další podkapitolou je uživatelský manuál, ve kterém jsou popsány parametry pro spuštění. A poslední podkapitola se věnuje popisu uživatelského rozhraní programu.

Kapitola 4 je zasvěcena souhrnu výsledků experimentování s programem. Poslední a závěrečná kapitola 5 rekapituluje celou diplomovou práci.

2 Zpracování obrazu

Vzhledem k tomu, že se celá tato práce zabývá zpracování obrazu považuji za vhodné nejprve definovat obraz samotný. Definice obrazu je velmi problematická vzhledem k subjektivitě vnímání daného abstraktního výrazu. V této práci však budeme využívat definici popsanou v [ZAR04] a to jako spojitou funkci dvou proměnných.

$$p = f(x, y) \tag{2.1}$$

V diskrétní oblasti si pak obraz představujme jako dvourozměrnou matici hodnot (rastr), jež specifikují barvu v daném bodě rastru. Těmto bodům říkáme pixely a jejich hodnoty jsou interpretovány skrze barevný model, jenž využívá obraz.

2.1 Barevné prostory

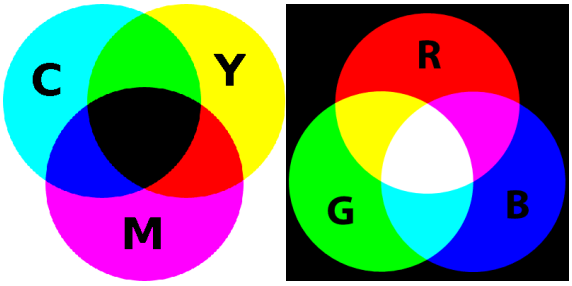
Barevný prostor definuje způsob interpretace a míchání barev v jejich diskrétní reprezentaci. Důvodů pro zavedení barevných prostorů je několik. Mezi nejdůležitější důvody patří redukce nadbytečných informací, optimalizace na kvalitu zobrazení, korelace dat a v neposlední řadě také intuitivnost míchání barev pro využití člověkem. V následujících podkapitolách jsou uvedeny některé z často používaných prostorů.

2.1.1 RGB

Nejpoužívanější barevný prostor k softwarové reprezentaci barev je prostor RGB, který odděleně reprezentuje intenzity R (červené), G (zelené), B(modré) části barvy. Počet barevných úrovní dané barvy je dán kvantizací daného formátu. U RGB se v současnosti nejvíce využívá 24 bitové reprezentace, kde je na každou složku vyhrazeno 8 bitů, takže všechny barevné složky jsou kvantovány 256 úrovněmi. Používají se i reprezentace s nerovnoměrným rozdělením kvantizačního kroku, např. při 16 bitové reprezentaci jsou složky R a B kvantizovány na 5 bitech a složka G na 6 bitech, toto nerovnoměrné rozdělení si můžeme dovolit vzhledem ke skutečnosti, že lidské oko je různě citlivé na jednotlivé složky a je dokázáno, že v modré a červené barvě rozpozná méně odstínů než-li v zelené.

Výsledná barva vznikne aditivním smícháním těchto tří složek. Na obrázku 2.1 je znázorněno aditivní a subtraktivní míchání základních složek modelu.

Aditivní míchání barev má uplatnění tam, kde vznikají barvy mícháním světla (např. monitor, dataprojektor atd.), zatímco v případech, kdy výsledná barva vzniká až od odrazu světla od materiálů, se využívá subtraktivní míchání barev, např. u tisku.

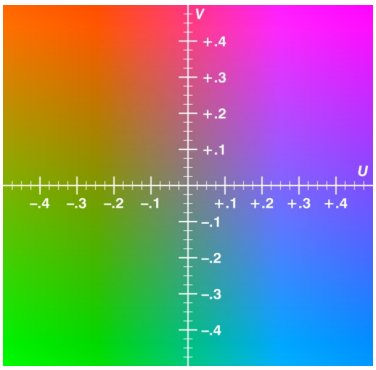


Obrázek 2.1: Ukázka subtraktivního a aditivního skládání základních složek

Ze vstupního senzoru tedy vystupuje obraz většinou v RGB barevném prostoru, v některých situacích je však výhodnější převést snímek do jiného barevného prostoru. Převody do rozdílných barevných prostorů se mohou využít také v situacích, kdy se chceme zbavit korelace dat. Tyto situace nastávají především v případech, kdy se do systému vloží blok, jenž využívá statistické metody, které pro své potřeby vyžadují informace z tohoto barevného prostoru.

2.1.2 Y'UV

V modelu YUV je oddělena jasová složka od barevné, čehož se využívá ve velkém počtu video standardů, např. PAL a NTSC. V [WIK01] se uvádí, že Y' reprezentuje jasovou složku v rozsahu 0-1. U a V jsou souřadnice v UV barevném čtverci v rozsahu -0.5 až 0.5. Blíže na obrázku 2.2. Pro úplnost dodávám, že tento obrázek je stav pro hodnotu Y'=0.5. Obrázek 2.2 byl převzat z [WIK01]



Obrázek 2.2: UV barevný čtverec

Podle [JEL07] převod z prostoru RGB na HSV lze provést pomocí následujícího vzorce 2.2 a pro převod z YUV na RGB lze využít vzorec 2.3.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.2}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.137 \\ 1.000 & -0.397 & -0.580 \\ 1.000 & 2.034 & 0.000 \end{bmatrix} \cdot \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \tag{2.3}$$

2.1.3 HSV

Barevný prostor HSV byl popsán v roce 1978[1] a je tvořen třemi složkami H, S a V. Tento model se využívá v počítačové grafice pro svůj intuitivnější charakter při míchání barev a odstínů. Dále se využívá ve zpracování videa pro oddělení jasové složky od složek barevných. Je použit ve spoustě video standardů.

Složka H (hue) odpovídá barevnému tónu a je vyjádřena jako úhel na standardním barevném kole a nabývá tedy hodnot 0° až 360°. Složka S (saturation) vyjadřuje sytost dané barvy tzn. poměr množství šedi k odstínu. Tato vlastnost se někdy nazývá též chroma a má rozsah hodnot 0 až 1.

Poslední složka V (value) je hodnota jasu. Jinými slovy složka určuje množství bílého světla v barvě a rozsah hodnot je také 0 až 1. Často se tento model vyjadřuje pomocí barevného kužele, kde složka H je nanesena jako úhel na podstavě kuželu, S představuje vzdálenost od osy tělesa a V znázorňuje výšku v kuželu.

Přepočet z prostoru RGB je proveden dle následujících vzorců 2.4, 2.5 a 2.6.

$$h = \begin{cases} nedefinovan\ ,\ jestliže\ max = min \\ 60^\circ \times \frac{g-b}{max-min} + 0^\circ,\ jestliže\ max = r\ a\ g \geq b \\ 60^\circ \times \frac{g-b}{max-min} + 360^\circ,\ jestliže\ max = r\ a\ g < b \\ 60^\circ \times \frac{g-b}{max-min} + 120^\circ,\ jestliže\ max = g \\ 60^\circ \times \frac{g-b}{max-min} + 240^\circ,\ jestliže\ max = b \end{cases} \tag{2.4}$$

$$v = max \tag{2.5}$$

$$s = \begin{cases} 0, & jestliže\ max = 0 \\ \frac{max-min}{max} = 1 - \frac{min}{max}, & v\ ostatních\ případech \end{cases} \tag{2.6}$$

1 Phd. Alvy Ray Smith

2.1.4 HSL

Model HSL je velmi podobný již popisovanému HSV s tím rozdílem, že HSL si lze představit spíše jako dva kužely přivracené k sobě, kde H a i S odpovídají zhruba stejnému významu jako u HSV, ale L jde od vrchního kuželu z hodnoty 1 (bíla) přes střed, kde se rovná 0.5(tedy šedé) postupně až k vrcholu spodního kuželu, kde je hodnota nulová.

Přepočet RGB na HSL se provádí dle následujících vzorců 2.7, 2.8 a 2.9.

$$h=\left\{\begin{array}{c} nedefinovan,\ jestliže\ max=min \\ 60^\circ\times\frac{g-b}{max-min}+0^\circ,\ jestliže\ max=r\ a\ g\geq b \\ 60^\circ\times\frac{g-b}{max-min}+360^\circ,\ jestliže\ max=r\ a\ g<b \\ 60^\circ\times\frac{g-b}{max-min}+120^\circ,\ jestliže\ max=g \\ 60^\circ\times\frac{g-b}{max-min}+240^\circ,\ jestliže\ max=b \end{array}\right\} \tag{2.7}$$

$$l=\frac{1}{2}(max+min) \tag{2.8}$$

$$s=\left\{\begin{array}{c} 0,\ jestliže\ l=0\ nebo\ max=min \\ \frac{max-min}{max+min}=\frac{max-min}{2l},\ jestliže\ 0<l\leq\frac{1}{2} \\ \frac{max-min}{2-(max+min)}=\frac{max-min}{2-2l},\ jestliže\ l>\frac{1}{2} \end{array}\right\} \tag{2.9}$$

Kde r,g a b jsou hodnoty barvy v RGB prostoru, max je největší hodnotou z těchto tří složek a min je nejmenší složkou.

2.1.5 Grayscale

Grayscale barevný model přepočítává barvy jednotlivých pixelů na odstíny šedi. Kde odstín šedi odpovídá jasu daného pixelu. Jas se z původního RGB obrázku získá pomocí následujícího vzorce.

$$G=0.299*R+0.587*G+0.114*B \tag{2.10}$$

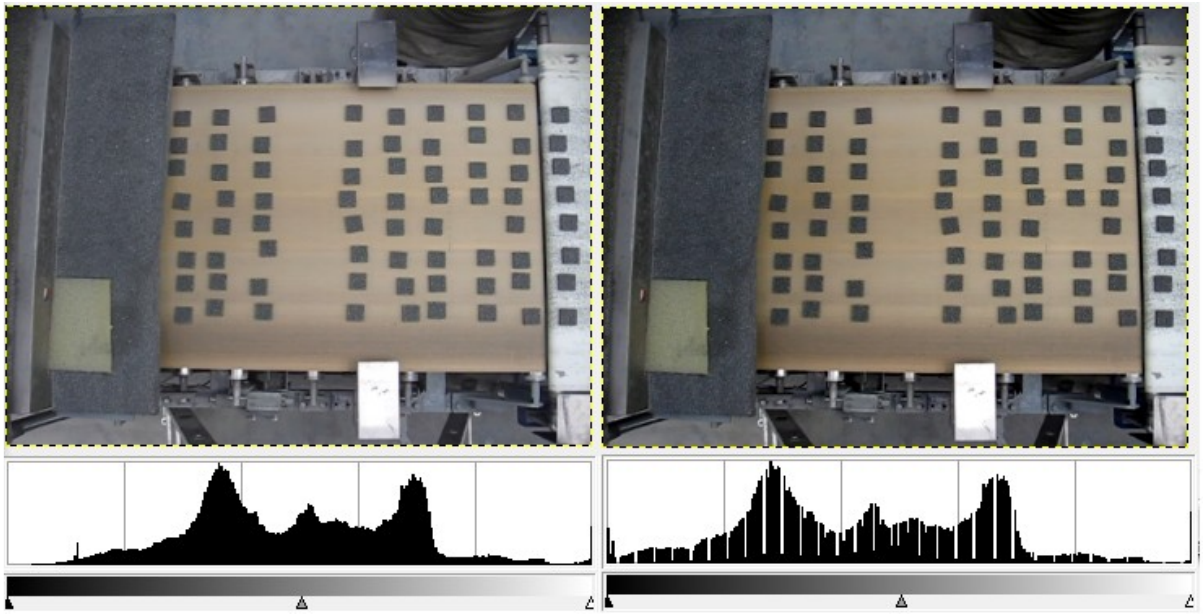
Kde R označuje červenou, G zelenou a B modrou složku pixelu, v rozsahu 0-1. Přepočítávací konstanty tohoto vzorce se mohou různě lišit podle implementace algoritmu, zde uvedené hodnoty jsou konstanty, které používá knihovna OpenCV, jež je použita pro implementaci systému. Každá barevná složka má jinou přepočítávací konstantu v závislosti na tom, jak je daná složka vnímána člověkem.

2.2 Ekvalizace histogramu

Vzhledem k měnícímu se osvětlení ve scéně, a tedy i měnící se dynamice, je vhodné provádět korekci dynamiky právě pomocí ekvalizace histogramu.

Obecně lze říci, že histogram je graf počtu výskytů všech jasových úrovní v obraze, kde hodnota jednotlivých sloupců odpovídá poměrné četnosti dané jasové úrovně v obraze. Součet všech sloupců je roven 1 v případě, že je tento histogram normalizován. Počet jasových intenzit je typicky 256, teoreticky je však možné vytvořit histogram s jakýmkoli rozsahem.

Transformací ekvalizace se rozumí roztáhnutí histogramu tak, aby efektivně pokryl celý rozsah intenzit, a tím byla dosažena větší dynamika obrazu. Na obrázku 2.3 je v levé části zobrazena testovací fotografie s průběhem jejího histogramu, vpravo se nachází obrázek po provedení ekvalizace. Tento obrázek byl vytvořen v programu GIMP.



Obrázek 2.3: Histogram před a po aplikace ekvalizace histogramu

Výpočet transformace ekvalizace je dána tímto vzorcem.

$$I' = \frac{I}{X \cdot Y} \sum_{i=I_0}^{I} H(i) \tag{2.11}$$

Kde I' je nová (ekvalizovaná) hodnota intenzity původní intenzity I, I_0 je nejnižší intenzita v původním obraze, X a Y jsou rozměry obrazu a funkce H(n) je funkcí neekvalizovaného histogramu.

2.3 Morfologické operace

Morfologické operace jsou nelineárními funkcemi nad binárním nebo šedotónovým obrazem. Dilatace a eroze vytváří základní stavební bloky pro složitější morfologické operace, jež jsou definovány podle následujících vztahů.

- Eroze objektu A podle strukturního elementu B. Používá se pro odstranění šumu ze vstupního obrazu.

$$A \ominus B = \{z \mid (B)_z \subset A\} \tag{2.12}$$

- Dilatace objektu A podle strukturního elementu B. Dilatace je využívána pro zaplnění děr v segmentech obrazu.

$$A \oplus B = \{z \mid (B)_z \cap A \neq 0\} \tag{2.13}$$

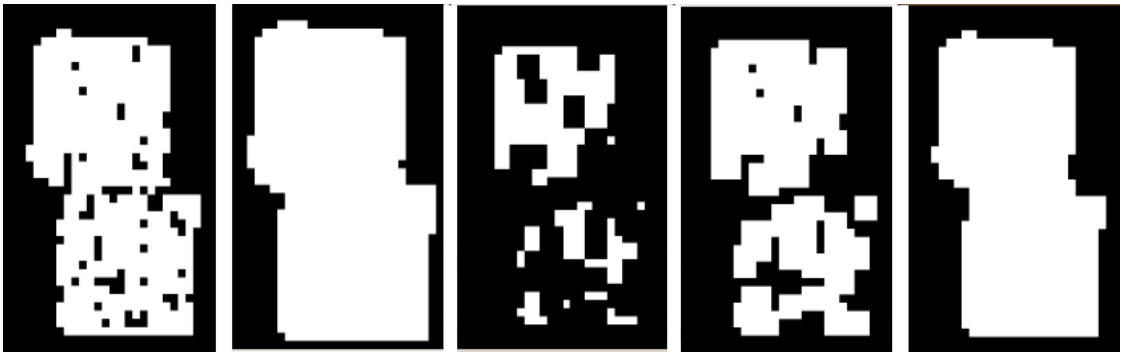
- Otevření objektu A podle B

$$A \circ B = (A \ominus B) \oplus B \tag{2.14}$$

- Uzavření objektu A podle B

$$A \bullet B = (A \oplus B) \ominus B \tag{2.15}$$

Kde A je vstupní obrázek a B je strukturní element, vyjádřený malým obrázkem. Mezi další morfologické operace patří skeletonizace, boundary detection (detektor okrajů) a mnoho dalších.
Na obrázku 2.4 vidíme výsledky morfologických operací na testovací fotografii. Tyto obrázky byly vygenerovány pomocí knihovny OpenCV.



Obrázek 2.4: Zleva: originál, dilatace, eroze, otevření, zavření

2.4 Filtrace obrazu

Filtrace šumu vstupního senzoru eliminuje chyby na snímacím zařízení, způsobené fyzikální nedokonalostí snímače a různými parazitními jevy, které jsou produkovány snímačem samotným nebo přenosovým kanálem (toto platí v případě, že z kamery vede analogový výstup, např. po koaxiálním kabelu).

Digitální kamery jsou často postihnuty tzv. barevným šumem, a to především za horších světelných podmínek.

2.4.1 Průměrovací filtr

Princip tohoto algoritmu je již z názvu poměrně jasný. Základem je provedení průměru určité oblasti v okolí daného pixelu. Matematicky se tato operace, pro velikost oblasti 3* 3, zapisuje následovně.

$$p(i,j)=\frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f(k,l) \tag{2.16}$$

Kde funkce $f(k,l)$ vrací hodnotu pixelu v rastru na pozici [k,l]. Častěji se však toto filtrování provádí pomocí konvoluce, kde se obraz konvoluje s následujícím průměrovacím jádrem. Výhodou tohoto přístupu je rychlost provedení filtrace, mezi nevýhody patří především nevhodnost filtru pro data, která jsou zatížena výstřelovým šumem. V takových případech se totiž průměrná hodnota spočítá jako hodnota vzdálená od většiny hodnot z daného okolí.

$$kernel=\begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$

2.4.2 Mediánový filtr

Stručný princip mediánu je možno shrnout následovně. Medián vybírá statisticky nejreprezentativnější člen posloupnosti čísel tak, že tyto prvky setřídí a zvolí prostřední prvek. Tento princip má výhodu oproti obyčejnému aritmetickému průměru v tom, že je omezen vliv extrémních hodnot.

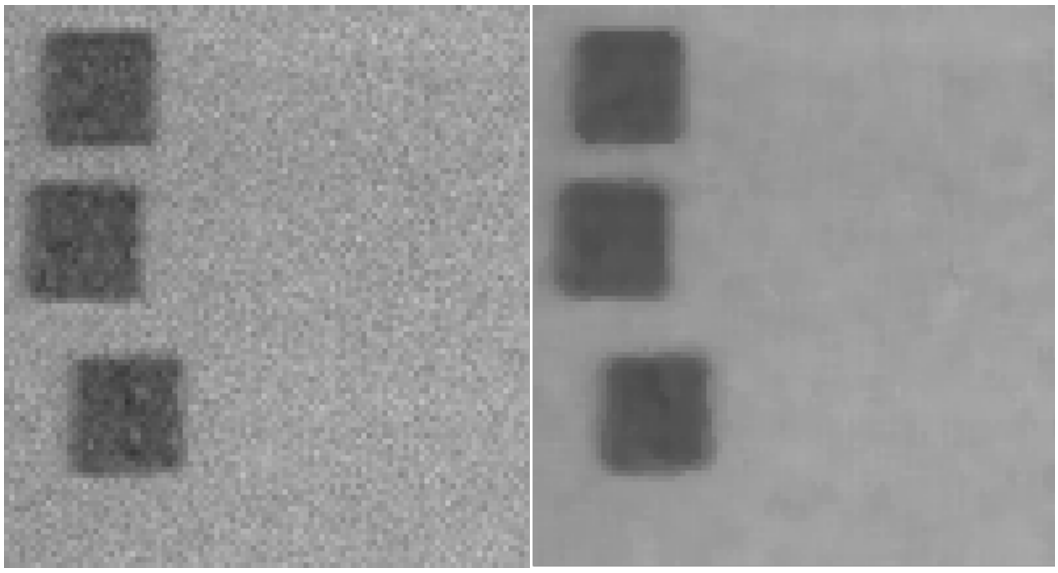
V diskrétní oblasti se medián může vyjádřit pomocí vzorce 2.17.

$$P(X \leq m) \geq \frac{1}{2} \wedge P(X \geq m) \geq \frac{1}{2} \tag{2.17}$$

Ve spojité pak.

$$\int_{-\infty}^{m} f(x)dx = \frac{1}{2} \tag{2.18}$$

Obrázek 2.5 ilustruje aplikaci mediánu na výřezu obrázku, kde se na levé straně nachází výřez zatížen šumem. Na pravé straně je pak zobrazen tentýž výřez po aplikaci mediánu.



Obrázek 2.5: Výřez obrázku před a po aplikaci mediánu.

2.5 Extrakce popředí

V tomto funkčním bloku se provádí segmentace obrazu na dvě vzájemně disjunktní oblasti, a to popředí a pozadí, kde popředím se rozumí ta část obrazu, která je neměnná vůči času, tento předpoklad samozřejmě platí jen tehdy, když vstupní obraz je snímán ze stacionárně umístěné kamery. Typicky se tedy popředím rozumí pohybující se objekty a pozadím neměnný prostor nacházející se za objekty.

2.5.1 Odečítání pozadí

Jednou z nejpoužívanějších metod je odečítání pozadí, pod tímto pojmem je skryt velmi jednoduchý princip, ve kterém je pozadí reprezentováno bitmapou pořízenou v situaci, kdy ve scéně před kamerou se nachází pouze pozadí bez žádného pohybujícího se objektu.

Samotné odečítání pak probíhá jako absolutní hodnota rozdílu každého pixelu vstupního obrazu s jeho ekvivalentním protějškem z bitmapy pozadí. Výsledkem je tedy opět bitmapa absolutních hodnot rozdílů, z tohoto obrázku získáme masku popředí pomocí binárního prahování.

Pro šedotónový obraz výpočet bitmapy probíhá podle následujícího vzorce 2.19.

$$f(x,y)=|i(x,y)-b(x,y)| \tag{2.19}$$

Kde i(x,y) je funkce intenzit originálního obrázku a b(x,y) je funkce intenzit obrázku pozadí. Nevýhoda této metody je neschopnost adaptace na přirozené změny pozadí (změna osvětlení, posunutí kamery). Mezi velké přednosti zajisté patří rychlost metody a také odolnost proti chybám u dlouho se nepohybujících objektů.

2.5.2 Modelování pozadí pomocí GMM

Z důvodů pochopení výše uvedené metody považuji za vhodné nejprve osvětlit následující pojmy.

2.5.2.1 Normální rozložení

Normální rozložení (též nazýváno Gaussovo rozložení) modeluje rozložení hustoty pravděpodobnosti dané náhodné veličiny. Toto rozložení je hojně používáno ve statistice a příbuzných oborech pro aproximaci pravděpodobnosti náhodných jevů.

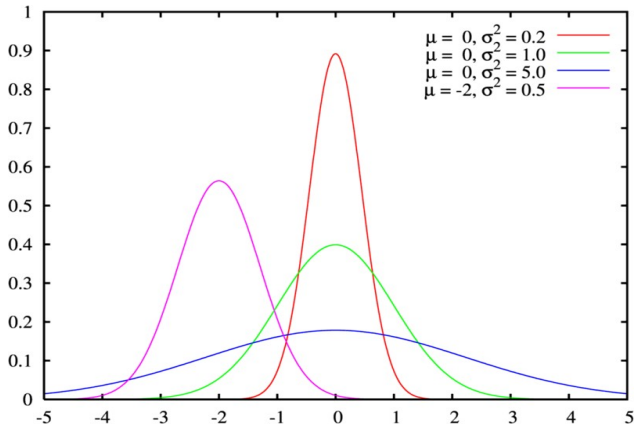
Gaussovo rozdělení je specifikováno pomocí parametrů střední hodnoty μ a rozptylu σ^2. Funkce hustoty pravděpodobnosti má pak tvar[2].

$$N(x,\mu,\sigma^2)=f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$
$$1=\int_{-\infty}^{\infty} f(x)dx \tag{2.20}$$

Distribuční funkce je vyjádřena následovně.

$$F(x)=\int_{-\infty}^{x}\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{-(t-\mu)^2}{2\sigma^2}}dt \tag{2.21}$$

Plocha pod grafem Gaussovy křivky je vždy rovna hodnotě 1 ,z čehož jasně vyplývá, že funkční hodnoty funkce rozdělení pravděpodobnosti nejsou v intervalu 0-1.



Obrázek 2.6: Ukázka Gaussových křivek s různými parametry.

Obrázek 2.6 byl převzat z[2] a jsou na něm vyobrazeny Gaussovy křivky s různými parametry.

Pokud je Gaussova křivka počítána nad více rozměry, je hustota pravděpodobnosti definována následovně.

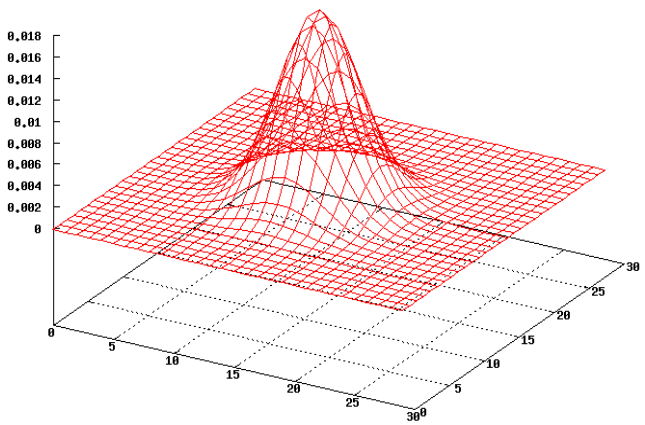
2 Zdroj cs.wikipedia.org/wiki/Normální_rozdělení

$$\mathcal{N}(\vec{x};\mu,\Sigma)=f(\vec{x})=\frac{1}{\sqrt{(2\pi)^s|\Sigma|}}e^{\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)} \tag{2.22}$$

Kde $\vec{x}=(x_{1,}x_{2,}x_{3,}\dots,x_s)$ je vektor vstupních hodnot, Σ je kovarianční matice a s je rozměr prostoru. Kovarianční matice zjednodušeně řečeno určuje směr natočení Gaussovy křivky mezi rozměry. Často se využívá tzv. diagonální kovarianční matice, která má hodnoty pouze na diagonále, Gaussova křivka je pak rovnoběžná s osami jednotlivých prostorů a tím pádem je výrazně zjednodušen výpočet funkce hustoty rozložení pravděpodobnosti.

$$f(\vec{x})=\prod_{i=1}^{k}\mathcal{N}(x_i,\mu_i,\sigma_i)=\prod_{i=1}^{k}\frac{1}{\sigma_i\sqrt{2\pi}}e^{\frac{-(x_i-\mu_i)^2}{2\sigma_i^2}} \tag{2.23}$$

Kde k je počet rozměrů. Následují obrázek 2.7 ilustruje rozložení hustoty pravděpodobnosti dvou náhodných veličin.



Obrázek 2.7: 2D gaussova křivka

2.5.2.2 Gaussian mixture model

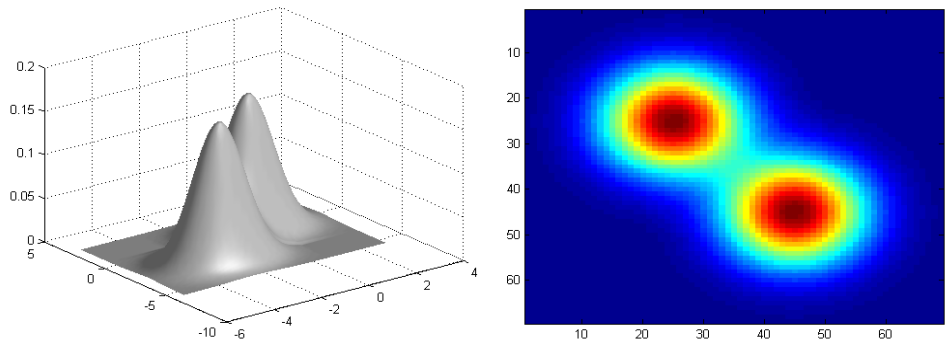
Gaussian mixture model (dále jen GMM) modeluje rozložení hustoty pravděpodobnosti pozorovaných veličin (např. barva, teplota, rozměr) pomocí směsi Gaussových křivek. Výsledná hustota rozložení je dána součtem jednotlivých Gaussových křivek a platí, že plocha (objem) pod grafem je rovna 1.

Výhoda tohoto modelu oproti modelu s jednou Gaussovou křivkou je v tom, že jedna Gaussova křivka nemusí vždy dobře pokrývat rozložení pravděpodobnosti daných jevů. Je tedy vhodné použít více křivek a výsledná funkce je pak součtem všech těchto křivek. U GMM na rozdíl od modelu s jednou Gaussovou křivkou přibývá další parametr váha α, který určuje tvar křivky.

Distribuční funkce má pak tvar.

$$f(\vec{x}) = \sum_{i=1}^{m} \alpha_i \mathcal{N}(\vec{x}, \mu_i, \Sigma_i)$$
$$\sum_{i=1}^{m} \int_{\infty}^{-\infty} \mathcal{N}_{\mu_i, \Sigma_i, \alpha_i}(\vec{x}) = 1 \tag{2.24}$$

Kde m je počet n-rozměrných Gaussových křivek, α_i váha dané Gaussovy křivky, $\vec{x}$ vektor vstupních hodnot, μ_i střední hodnota křivky a Σ_i je kovarianční matice. GMM je často n-dimenzionální, kde počet dimenzí odpovídá počtu příznaků. Aby tyto modely mohly správně fungovat, je potřeba provést inicializaci hodnot modelu. To se provádí typicky natrénováním na trénovacích datech.



Obrázek 2.8:Ilustrační obrázek 2D GMM

Na obrázku 2.8 je vpravo ve 3D prostoru znázorněna hustota pravděpodobnosti daného dvojrozměrného modelu. Vlevo se nachází vyjádření totožného modelu ve 2D reprezentaci. Z ilustračního obrázku vyplývá, že je model složen ze dvou dvojrozměrných Gaussových křivek. Obrázek byl vygenerován v programu Matlab

2.5.2.3 Model pozadí založený na GMM

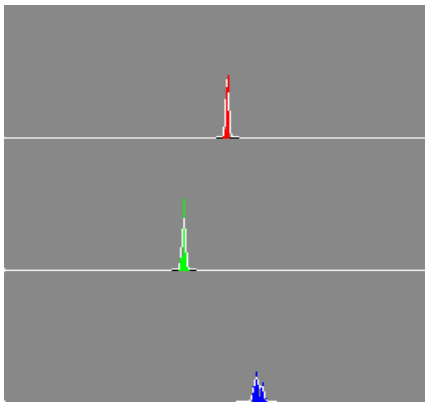
V modelu pozadí založeném na GMM je každý pixel, popřípadě blok pixelů (zrychlení metody), modelován pomocí výše uvedeného GMM, jenž udává rozdělení hustoty pravděpodobnosti dané barvy v místě pixelu.

Základním principem je modelování barevného histogramu tímto modelem, přičemž je vhodné, aby model měl minimálně dvě Gaussovy křivky. Jednu pro modelování skutečného pozadí a druhou pro modelování průchodu některého objektu.

Tento model může být vytvořen dopředu, pak se tomuto modelu říká statický. V takovém případě se model jednou natrénuje a dále se již parametry jednotlivých sub-modelů nemění.

Nicméně používanější variantou je však dynamická verze algoritmu, kdy se v průběhu zpracování dynamicky přizpůsobují parametry modelů a tím se stává celkový model adaptabilnější

vůči změnám okolního prostředí. Většinou se v takových případech definuje učící konstanta, která určuje míru rychlosti adaptace na změnu.



Obrázek 2.9: Natrénovaný model na barevný histogram

Na obrázku 2.9 je znázorněn naučený trojrozměrný model GMM, který je složen ze dvou Gaussových křivek. Učení tohoto modelu bylo provedeno pomocí algoritmu expectation maximization z knihovny OpenCV.

2.5.3 Extrakce pomocí prahování

Threshold, neboli prahování, patří k jedné z nejprimitivnějších segmentačních metod. V principu jde vlastně o jednoznačné zobrazení, které každému vstupnímu bodu z oboru odstínů šedi přiřadí binární hodnotu 1 nebo 0 podle daných kritérií. Těmito kritérii jsou úrovně jasu (prahy), pod kterými je výstupní hodnota rovna 0 a nad nimi 1.

Ze vstupního obrazu s 256 odstíny šedi se vyrobí výstupní obraz s pouze dvěma hodnotami tzv. binární obraz. Následující vzorec 2.25 udává předpis pro výpočet thresholdu s jedním prahem.

$$f(c)=\begin{Bmatrix} A|c<práh \\ B|c\geq práh \end{Bmatrix} \tag{2.25}$$

Kde A a B jsou výstupní hodnoty, v našem případě 0 a 255.

Varianta s dvěma prahy.

$$f(c)=\begin{Bmatrix} A|c>práh_1 \wedge c<práh_2 \\ B|c\leq práh_1 \vee c\geq práh_2 \end{Bmatrix} \tag{2.26}$$

Nevýhoda této metody spočívá v problému nastavení vhodného prahu, přičemž nastavení je založeno na důkladné znalosti podmínek v daném místě detekce.

2.5.3.1 Adaptivní threshold

Vzhledem k tomu, že ve vstupním obraze mnohdy nebývá konstantní osvětlení, popřípadě se v něm vyskytují různé stíny či barevné přechody, není snadné, nebo dokonce není vůbec možné, najít jeden optimální práh pro celý obraz.

Vhodnější metodou pro takové případy je adaptivní prahování, v němž se práh určuje v kontextu okolí bodu pomocí lokálního histogramu tohoto okolí. Níže uvedený vzorec udává výpočet adaptivního thresholdu.

$$f(c)=\begin{Bmatrix} A|c<H(c,w)[i] \\ B|c\geq H(c,w)[i] \end{Bmatrix} \tag{2.27}$$

Kde c je počítaný pixel obrazu, w je šířka okna, nad kterou se počítá histogram, a i označuje index prvku z histogramu, jenž se prohlásí za práh. Postup výpočtu je tedy následující. Nejprve se kolem daného bodu spočítá histogram, přičemž velikost okolí je daná parametrem w, poté se vezme i-tý prvek histogramu a je použit pro prahování.

2.6 Extrakce pomocných příznaků

Dalším krokem pro rozpoznávání objektů v obraze je výpočet pomocných příznaků, jež se dají následně použít pro usnadnění detekce a rozpoznání objektů v obraze a jejich následné validace. Typickými pomocnými příznaky jsou hrany, jejich orientace a tloušťka, barevné histogramy, textury a mnoho dalších.

2.6.1 Detekce hran

Nalezení hran v obrázku je velmi důležitým krokem ve zpracování obrazu a nalezené hrany jsou velmi hodnotným zdrojem dat pro následné rozpoznávání objektů v obraze. Nejprve by bylo vhodné si říci, co rozumíme hranou v obrazových datech. V nejideálnějším případě je hrana dána vysokým rozdílem jasu ve svém okolí a hranu tedy popisuje rychlost změny gradientu jasu a její směr ψ .

- Gradient

$$\nabla f(x,y)=\left(\frac{\partial f}{\partial x},\frac{\partial f}{\partial y}\right) \tag{2.28}$$

- Velikost gradientu

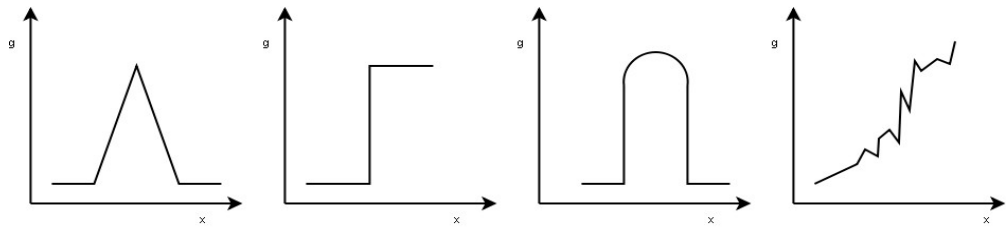
$$\|\nabla f(x,y)\|=\sqrt{\left(\frac{\partial f}{\partial x}\right)^2+\left(\frac{\partial f}{\partial y}\right)^2} \tag{2.29}$$

- Směr gradientu

$$(\sin\psi,\cos\psi)=\left(\frac{\partial f}{\partial x},\frac{\partial f}{\partial y}\right) \tag{2.30}$$

Je však zřejmé, že nejen skutečné hrany objektů budou ve vstupním obrázku vytvářet hrany. Například vržené stíny jsou zdrojem nepříjemných hran. Tyto hrany sice mají přímou souvislost s hranicí původního objektu, která není triviální, ale pro rozpoznání pravé hrany objektu jsou tyto vržené stíny spíše komplikací. Na druhou stranu pokud barva objektu splývá s pozadím, nevzniknou hrany žádné. Dalším zdrojem falešných hran mohou být odlesky světla od hladkých materiálů a v neposlední řadě také sama textura objektů.

Pokud se tedy hrana definuje jako náhlá změna jasu, pak maximum její první derivace určuje směrnici kolmou k této hraně. Výpočet této směrnice lze aproximovat pomocí konvoluce, jež se provede se vstupním obrazem a konvolučním jádrem.



Obrázek 2.10: Různé hranové profily

Na obrázku 2.10 jsou znázorněny různé průběhy gradientu. První tři případy průběhu jsou značně ideální a mohli bychom je najít typicky u syntetické grafiky např. vektorové grafiky. Poslední průběh znázorňuje zašumělou hranu, která nejvíce odpovídá reálným průběhům gradientu.

2.6.1.1 Metody založené na první derivaci

První derivaci lze aproximovat konvolucí původního obrazu s konvolučním jádrem. Existuje spousta různých konvolučních jader určených k různým účelům. Na ukázku uveďme několik nejznámějších, které jsou založeny na výpočtu maxima první derivace.

- Robertsův operátor detekuje hrany v kolmých směrech. Nevýhodou je vysoká citlivost na šum, vzhledem k malému okolí.

$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

- Prewittové operátor 3x3. Slouží k detekci hran v y ose, pokud se vyžaduje detekce v x-ose stačí matici analogicky upravit. Popřípadě se dají zkonstruovat i jádra pro šikmé směry.

$$\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

- Prewittové operátor 5x5. Tento operátor je odolnější vůči šumu než jeho předchozí varianta, přičemž platí, že čím větší matice, tím větší odolnost proti šumu, ale na druhou stranu zaniká více slabých hran.

$$\begin{pmatrix} -2 & -2 & -2 & -2 & -2 \\ -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \end{pmatrix}$$

- Sobelův operátor počítá hodnotu gradientu hrany. Matice uvedená níže je Sobelův operátor pro kolmé hrany. Rovněž je možné sestrojit jádro pro vodorovný směr a šikmé směry.

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

- Kirschův operátor

$$\begin{pmatrix} -5 & -5 & -5 \\ 3 & 0 & 3 \\ 3 & 3 & 3 \end{pmatrix}$$

- Robinsonův operátor

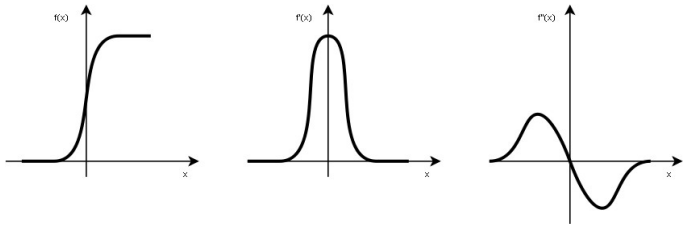
$$\begin{pmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Na následujícím obrázku 2.11 je nalevo vyobrazen originální obraz převedený do odstínů šedi. Na pravé straně obrázku je pak odezva Prewittové operátoru 5*5.

*Obrázek 2.11: Ilustrační obrázek demonstrující Prewittové operátor 5*5*

2.6.1.2 Metody založené na druhé derivaci

Kromě hledání maxima první derivace lze využít i nalezení průchodu 0 druhé derivace (tzv. Zero-crossings). Na obrázku 2.12 je možno vidět průběh první a druhé derivace vzhledem k průběhu signálu. Zatímco u první derivace je jasně vidět, že v místě hrany leží maximum funkce, u druhé derivace je tomu přesně naopak a v místě hrany funkce prochází nulou.



Obrázek 2.12: Vztah signálu vůči své první a druhé derivaci

Laplacián je stručně řečeno všesměrová druhá derivace a oproti gradientu je to skalární hodnota, přicházíme tedy o informaci o směru. Vzhledem k tomu, že je laplacián druhou derivací, má nulovou hodnotu v místě středu hrany, tedy tam, kde je velikost gradientu maximální.

$$\Delta = \nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2} \tag{2.31}$$

Druhou derivaci lze v diskrétním prostoru aproximovat konvolucí prvních derivací. Takto vypadá např. jeho maticové vyjádření.

$$\Delta = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

2.6.2 Detekce rohů

Detekování rohů v obraze je jedním z nejdůležitějších zdrojů informací popisujících geometrii detekovaného objektu. Proto v této kapitole budou popsány některé nejznámější detektory rohů.

2.6.2.1 Moravec algoritmus

Dle [WIK02] je Moravec[3] algoritmus jeden z prvním algoritmů zabývajících se detekcí rohů podle velké míry shody okolí se samotným bodem. Algoritmus testuje každý pixel vstupního obrazu a zkouší zda není potenciálním rohem. Algoritmus lze popsat následovně.

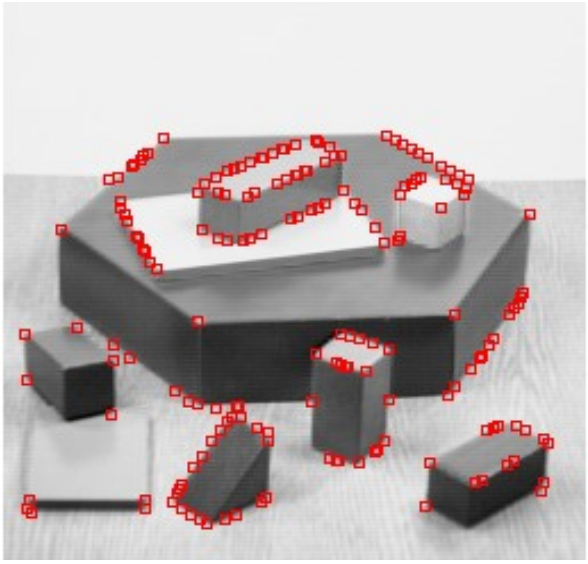
- Nad každým bodem se umístí čtvercová maska
- Touto maskou se pohybuje do všech 8 směrů.
- Pro všechny posunutí se vypočítá odlišnost podle vzorce 2.32

$$V_{u,v}(x,y)=\sum_{a,b\in window}(I(x+u+a,y+v+b)-I(x+a,y+b))^2 \tag{2.32}$$

- Jako výstupní hodnota pro daný pixel se vyhodnotí minimum ze všech směrů

$$C(x,y)=min(V_{u,v}(x,y)) \tag{2.33}$$

- Provede se prahování výstupní mapy



Obrázek 2.13: Detekce rohů pomocí algoritmu Moravec

Na obrázku 2.13 jsou rámečky vyznačeny nadetekované rohy. Můžeme si zde všimnout, že má metoda problémy s detekcí rohů na šikmých hranách, kde vznikají falešné rohy.

3 Algoritmus byl vyvinut Hansem Moravcem v roce 1977 v rámci výzkumu robotického vidění. Mimo jiné je Moravec také tvůrcem pojmu ROI(region of interest) a spisovatelem sci-fi literatury.

2.6.2.2 Wang Brady algoritmus

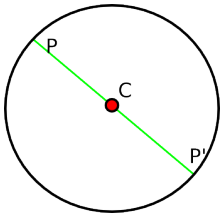
Wang Brady algoritmus detekuje rohy jako náhlou změnu křivosti křivky. Rohy se tedy detekují v místech, ve kterých dochází k prudké změně směru křivky. Výpočet těchto bodů se počítá podle následujícího vzorce.

$$C = \nabla^2 I - c\,|\nabla I|^2 \tag{2.34}$$

Kde c určuje citlivost detektoru na změnu směru křivky. Je také doporučeno použití guassovského vyhlazení pro redukci šumu. Toto vyhlazení ovšem přináší určitou nepřesnost u ostrých úhlů. Toto lze ovšem dodatečně korigovat.

2.6.2.3 Trajkovic algoritmus

Existují dvě verze algoritmu, tedy Trajkovic4[4] a Trajkovic8. Rozdíly mezi nimi jsou především v různých velikostech okolí testovaného pixelu. Definice rohu je stejná jako v případě již popsaného Moravec algoritmu.



Obrázek 2.14: Definice bodu a jeho okolí

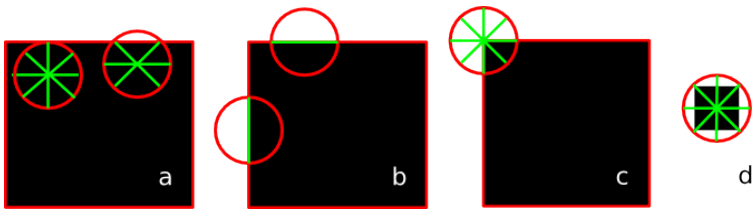
Míra rohovitosti se vypočítá následovně.

$$C(x,y) = min((I_P - I_C)^2 + (I_{P'} - I_C)^2), \forall P, P' \tag{2.35}$$

Kde I_C je intenzita v centru kružnice, I_P je intenzita v bodě P a $I_{P'}$ je intenzita v bodě P'. V této chvíli nastávají čtyři možnosti viz. obrázek 2.15 .

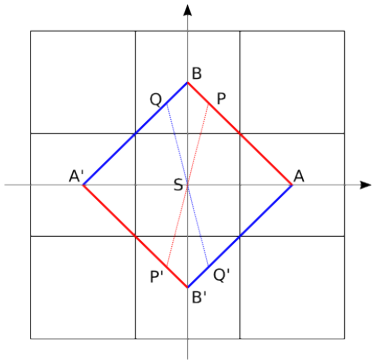
a) Zkoumaný bod se nachází ve vnitřní oblasti, jestliže platí $I_C \approx I_P \approx I_{P'}$ pro všechny spojnice z P do P' přes C.
b) Bod se nachází na hraně v případě, že existuje pouze jedna spojnice mezi P do P' přes C přičemž musí platit $I_C \approx I_P \approx I_{P'}$.
c) Bod se nachází na rohu, jestliže pro všechny úsečky z P do P' platí, že nejsou uzavřené přes C.
d) Izolovaný bod je definován jako bod. pro který platí, že I_C je různé od všech I_P a $I_{P'}$.

4 Algoritmus společně sestavili Miroslav Trajkovic a Mark Hedley v roce 1998.



Obrázek 2.15: a)bod uvnitř oblasti b)bod na hraně oblasti c)bod na rohu oblasti d)izolovaný bod

Na obrázku 2.15 vidíme jednotlivé případy uvedené v předchozích bodech. Z výše uvedeného vyplývá, že tento detektor bude relativně dobře fungovat na nezašumělých obrazech. Tento handicap je možno eliminovat pomocí filtrace (např. medián) nebo gaussovského rozmazání. Z kruhové reprezentace je potřeba při samotném výpočtu přejít na reprezentaci v rastru. Tento převod je potřeba provést pomocí interpixelové aproximace.



*Obrázek 2.16: Aproximace pixelů do okna 3*3*

Výpočty změn intenzit v horizontální a vertikálním směru jsou dány jednoduchým vztahem 2.36.

$$\begin{aligned} C_{SIMPLE} &= min(r_A, r_B) \\ r_A &= (I_A - I_S)^2 + (I_{A'} - I_S) \\ r_B &= (I_B - I_S)^2 + (I_{B'} - I_S) \end{aligned} \tag{2.36}$$

Výpočet přímek, které prochází středem kružnice, avšak neprochází osami souřadného systému, je poněkud složitější. Lineární aproximace se provádí podle následujícího vztahu.

$$\begin{aligned} C_{INTERPIXEL}(x, y) &= min(r_1(x), r_2(x)) \\ r_1(x) &= A_1 x^2 + 2B_1 x + C \\ r_2(x) &= A_2 x^2 + 2B_2 x + C \\ B_1 &= (I_B - I_A)(I_A - I_S) + (I_{B'} - I_{A'})(I_{A'} - I_S) \\ B_2 &= (I_B - I_{A'})(I_{A'} - I_S) + (I_{B'} - I_A)(I_A - I_S) \\ C = r_A \quad , \quad A_1 &= r_B - r_A - 2B_1, \quad A_2 = r_B - r_A - 2B_2 \end{aligned} \tag{2.37}$$

2.6.2.4 Susan algoritmus

Susan (Smallest Univalue Segment Assymilating Nucleus) algoritmus detekuje rohy pomocí porovnání daného bodu s jeho okolím, kde okolí je definované pomocí kružnice. Výhoda SUSAN metody je její odolnost vůči šumu a rychlost výpočtu, avšak mezi její nevýhody patří špatná detekce u souběhu dvou šikmých hran. Obvyklý rádius masky bývá 3.4 pixelu, kruh o tomto rádiu pojme cca 37 pixelů.

Maskou se postupně projde celý obraz, pixel po pixelu. Podle [WIK02] se v každém bodě provede následující porovnávací funkce.

$$c(\vec{m}) = e^{\frac{(I(\vec{m}) - I(\vec{m}_0))^6}{t}} \tag{2.38}$$

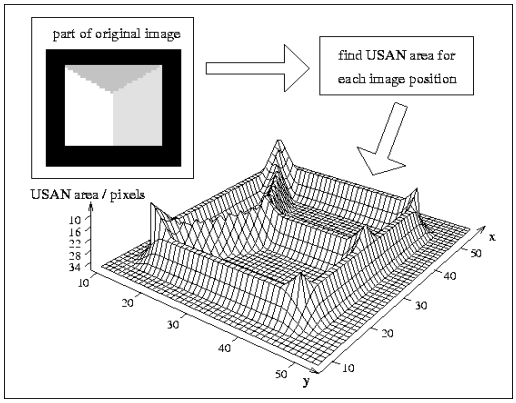
Kde t určuje šířku, $\vec{m}$ jsou pixely dané maskou, $\vec{m}_0$ je nucleus (zkoumaný pixel) a velikost exponentu byla zjištěna empiricky. Oblast USAN je popsána vzorcem.

$$n(M) = \sum_{\vec{m} \in M} c(\vec{m}) \tag{2.39}$$

Kde n dává počet pixelů v masce M se stejnou nebo podobnou intenzitou jako zkoumaný pixel. Samotný operátor USAN je definován následovně.

$$R(M) = \begin{cases} g - n(M) & \textit{jestliže } n(M) < g \\ 0 & \textit{v ostatních případech} \end{cases} \tag{2.40}$$

Kde g se nazývá geometrický threshold. Operátor vrací kladný výsledek pouze v případě, že je oblast dostatečně malá. Parametr t určuje, jak moc podobné body se počítají do USAN oblasti a parametr g určuje minimální velikost oblasti. Pro detekci rohů je potřeba ještě provést dva kroky. Zaprvé nalezení těžiště oblasti (v místě rohu je těžiště velmi vzdáleno od nucleidu) a zadruhé provést otestování, zda všechny body na spojnici z nucleidu do těžiště, a dál na okraj USAN masky, jsou právě v USAN oblasti.



Obrázek 2.17: 3D vizualizace odezvy algoritmu SUSAN na testovací obrázek

2.6.3 Texturní příznaky

Často se pro detekci objektů ve scéně využívají texturní příznaky, které určitým způsobem definují vlastnosti textury dané třídy objektů. Texturou rozumíme opakující se strukturu obrazových primitiv (též nazývaných texely) ale mohou to být i čáry, geometrické obrazce apod. Metody pro analýzu textur se dají rozdělit na dva hlavní směry, a to podle principu popisu textury na strukturní, kde je textura popsána podle její struktury gramatikou či polygonální sítí, a statistické, ve kterých se textura popisuje podle statistických vlastností intenzit pixelů ve vzoru.

V této práci se budu výhradně zabývat druhou jmenovanou skupinou metod, jelikož první výše zmíněná skupina metod je nevhodná k řešení problémů tématu mé diplomové práce. Statistické metody reprezentují danou texturu vektorem příznaků.

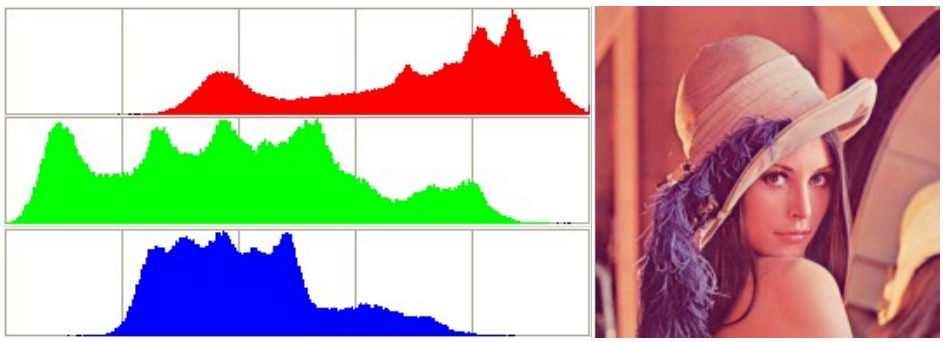
2.6.3.1 Barevný histogram

Textura se dá vyjádřit vektorem hodnot, kde složky vektoru představují sloupce barevného histogramu specifického pro danou třídu textury. Texturní příznaky postavené na barevných histogramech mají tu výhodu, že jsou invariantní vůči rotaci, translaci i změně projekce ve 3D prostoru. Bohužel pokud se jedná o barevný histogram provedený nad barevným prostorem RGB, jsou tyto příznaky citlivé na změnu osvětlení objektu.

Barevný histogram nad obrazem I je možné vyjádřit jako vektor hodnot $\vec{h}=(h_{1,}h_2,\ldots,h_n)$, kde prvek vektoru h_i udává četnost barvy i v obraze I. Musí nadále platit následující vztah.

$$\sum_{i=0}^{n} h_i = w_I * h_I \tag{2.41}$$

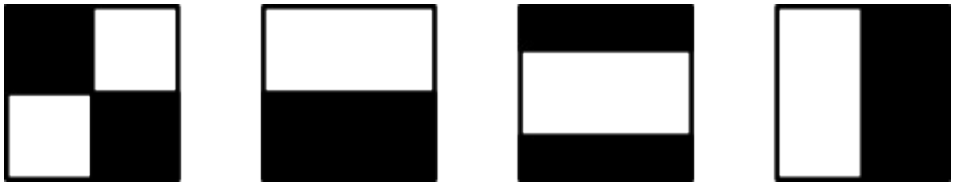
Kde w_I je šířka a h_I je výška obrázku I. Pokud by měl obrázek více kanálů, je pak vektor $\vec{h}$ vícerozměrný.



Obrázek 2.18: Testovací obrázek a jeho barevný histogram

2.6.3.2 Haarovy příznaky

Haarovy příznaky jsou typickým zástupcem příznaků, které se používají v metodách statistického rozpoznávání se slabými klasifikátory (např. AdaBoost). Haarovy příznaky jsou definovány pomocí banky frekvenčních filtrů (Haarových bází) a textura je tedy určena vektorem příznaků, kde jednotlivé prvky vektoru jsou odezvy segmentu na jednotlivé báze, kde báze jsou vlastně konvolučními jádry



Obrázek 2.19: Příklady Haarových bází

Na obrázku 2.19 je ilustrace příkladů Haarových bází. Velikou výhodou těchto příznaků je jejich potenciální počet, kterým se dá popsat segment obrazu. Toto je typicky využito v již řečeném klasifikátoru AdaBoost, který vyžaduje velkou spoustu vstupních příznaků k dosažení solidního výsledku rozpoznávání.

Odezvy těchto filtrů se počítají podle následujícího vzorce 2.42.

$$f(x)=\sum_{w\in W} x(w)-\sum_{b\in B} x(b) \tag{2.42}$$

Kde W je množina pixelů pod bílou částí báze a B je množina pixelů pod černou částí báze. Další pozitivum těchto příznaků je rychlost výpočtu odezvy pomocí optimalizace předpočítáním integrálního obrazu. Integrální obraz má stejnou velikost jako obraz originální. V každém svém bodě udává součet pixelů v obdélníku od aktuálního bodu do levého horního rohu. Tímto přístupem se rovněž zajistí konstantní čas výpočtu jakkoliv veliké obdélníkové oblasti v obraze, na rozdíl od obyčejného sekvenčního algoritmu součtu obecně jakkoliv velké obdélníkové oblasti, přičemž čas výpočtu algoritmu je přímo úměrný velikosti oblasti, nad kterou se počítá.

Vzorec 2.43 demonstruje matematickou reprezentaci integrálního obrazu.

$$ii(x,y)=\sum_{x'\leq x,\ y'\leq y} i(x',y') \tag{2.43}$$

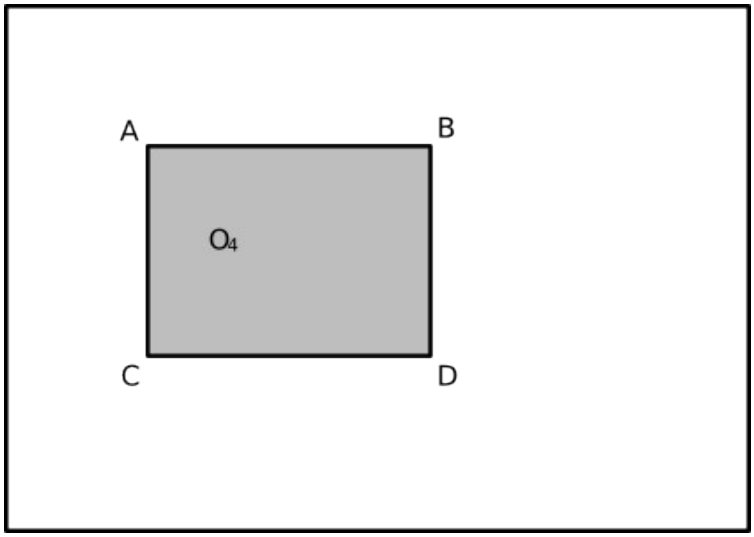
Výpočty integrálního obrazu se ovšem efektivněji počítají pomocí následujících vzorců.

$$r(x,y)=r(x,y-1)+i(x,y)$$
$$ii(x,y)=ii(x-1,y)+r(x,y) \tag{2.44}$$

Výpočet obdélníkové oblasti, která nemá levý horní roh v bodě [0,0], se počítá vzorcem 2.45 .

$$O_4=ii(D)+ii(A)-(ii(C)+ii(B)) \tag{2.45}$$

Kde velká písmena A,B,C a D označují souřadnice v integrálním obraze a O_4 je oblast, ve které se počítá součet pixelů viz. obrázek 2.20 .Pomocí tohoto vzorce je tedy možné spočítat sumu pixelů nad jakkoliv velkou oblastí obrazu v konstantním čase.



Obrázek 2.20: Ilustrace integrálního obrazu

2.7 Segmentace obrazu

Segmentace obrazu je specializace algoritmů obecně se zabývajících shlukováním. Shlukováním se obecně rozumí princip, který ze vstupních dat, ve kterých jsou oblasti, jež mají určitou logickou souvislost nebo topologickou sounáležitost, vytvoří souvislou oblast, jež se pak dá chápat jako celek a lze ji popsat souhrnným údajem.

Podle [SPA07] se shlukování používá především k těmto aplikacím.

- Segmentace obrazu, která rozdělí obraz na jednotlivé bloky ucelených oblastí dle příslušností k daný třídám příznaků (např. podle informací o jasu, barvě, textuře, hustoty hran apod.).
- Dolování v datech, kde se shlukování používá k automatickému rozpoznání shluků a rozdělení na třídy, kde nebylo předem dáno kolik tříd se v prostoru vyskytuje. Informaci o počtu tříd typicky v těchto případech nemáme a je ji potřeba automaticky zjistit.

V aplikacích pro detekci pohybujících se objektů na jedoucím páse bude shlukovací algoritmus provádět segmentaci obrazu na jednotlivé objekty.

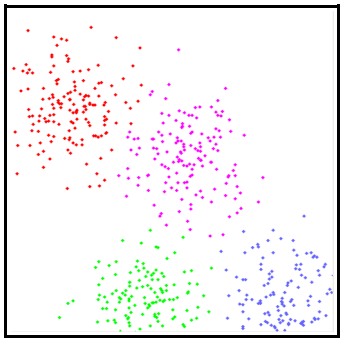
2.7.1 K-means

Algoritmus k-means rozděluje vstupní vektory do k shluků a každý shluk je definován svým těžištěm. Příslušnost jednotlivých dat ke shlukům určuje právě vzdálenost dat od jejich těžišť.

Podle [LOR07] je metoda K-means založena na tomto algoritmu.

- Náhodně se umístí do obrazu k nových shluků.
- Spočítá se nové těžiště shluků při tomto rozložení.
- Přiřadí se každý objekt do shluku, jehož těžiště je nejbližší.
- Pokud při přiřazení proběhla změna, pokračuje se na kroku 2, jinak se skončí.

Tento algoritmus je poměrně efektivní, na druhou stranu potřebuje specifikovat počet shluků, definovat těžiště a špatně zpracovává zašumělá data. Více o metodě k-means v [SPA07] . Následující obrázek 2.21 byl vygenerován z demonstrativního příkladu v knihovně OpenCV.



Obrázek 2.21: Rozdělená data do shluků pomocí metody k-means.

2.7.2 Mean-Shift

Data náležející do stejné třídy ovšem někdy vytváří shluky, které by svým tvarem byly předchozí metodou K-means klasifikovány chybně. Jedná se o shluky, jež nemají hustotu bodů centrovánu kolem svého středu ale vytvářejí podlouhlý shluk. Typický příklad takových shluků je na obrázku 2.22 .

Metoda mean-shift je založena na předpokladu, že data nabývají na hustotě směrem ke středu shluku. Dle [SPA07] se lokální hustota dá vyjádřit vzorcem.

$$f(x)=\frac{1}{Nh^d}\sum_{i=1}^{N} k\left(\frac{\|x-x_i\|}{h}\right) \tag{2.46}$$

Kde h je šířka kernelu a x1...n jsou vzorky a kernel k je vyjádřen jako.

$$k(x)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \tag{2.47}$$

A maxima jsou v místech, kde se gradient (první derivace) rovná 0.

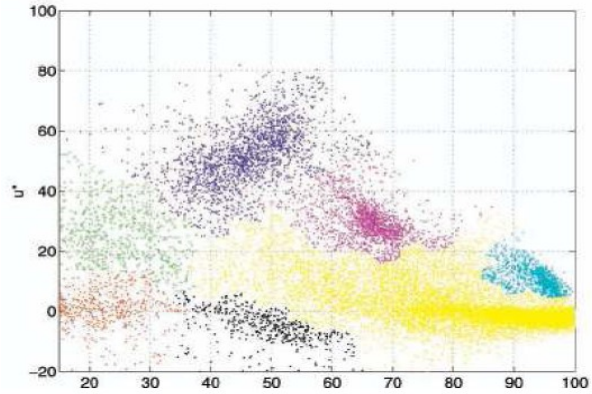
$$\nabla f(x)=0 \tag{2.48}$$

Stručný postup výpočtu je možno shrnout následovně.

- V každém vzorku se vypočítá lokální maximum funkce, do kterého algoritmus dokonvergoval, a tato hodnota se zapamatuje.
- Ty vzorky, které dokonvergují do stejného maxima, prohlásíme za shluk.

Jinými slovy se metoda posunuje postupně směrem k větší hustotě bodů. Detailnější popis algoritmu v [DOU07] následující obrázek 2.22 byl převzat z [SPA07].

Na obrázku 2.22 můžeme vidět výhodu této metody, a to schopnost najít i podlouhlé shluky na rozdíl od k-means, která detekovala spíše kruhové shluky.

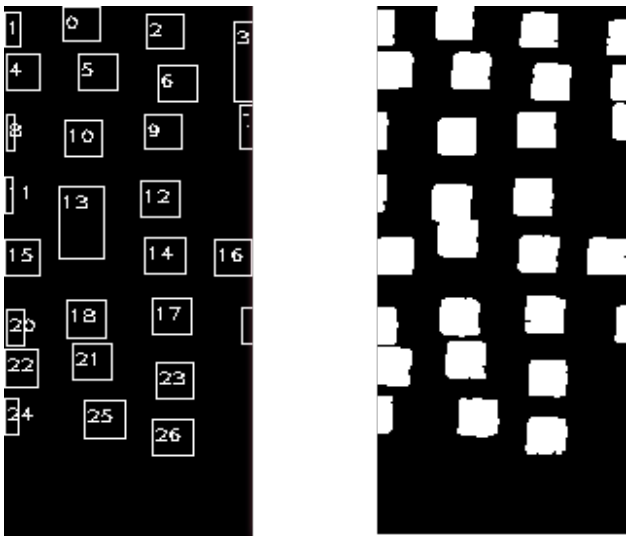


Obrázek 2.22: Rozsegmentovaná data metodou MeanShift

2.7.3 Connected Component Labeling

Metoda Connected Component Labeling (dále CCL) je také často označovaná jako metoda semínkového vyplňování nebo obarvování. Metoda patří k základním shlukovacím metodám a je založena na primitivním principu rozpoznání jednotlivých shluků.

Shluk je zde definován jako souvisle vyplněná oblast. Oblastí se rozumí region s podobnou intenzitou jasu pixelů, popřípadě lze použít jiný příznak. Spojitost oblastí se může definovat podle okolí 4, 8 nebo n bodů. Detekce probíhá rekurzivně nebo pomocí dvou průchodů obrazem.



Obrázek 2.23: Vlevo obálky nadetekovaných blobů. Vpravo vyprahovaný obrázek

Na obrázku 2.23 vlevo je vidět výsledek metody CCL nad obrázkem vpravo. Všimněme si, že každé nadetekované oblasti přísluší jedinečný číselný identifikátor (štítek). Mezi základní přednosti tohoto algoritmu patří rychlost detekce blobů.

Dvouprůchodový algoritmus pracuje následovně. V prvním průchodu se každému nenulovému pixelu, v případě že existuje nenulové okolí, přiřadí hodnota podle jeho sousedních pixelů takto.

- Pokud jsou okolní body součástí pozadí, přiřadí se danému pixelu nová hodnota (štítek,barva).
- Jeli právě jeden ze sousedů nenulový, přiřadí se danému pixelu hodnota tohoto souseda.
- Je-li více sousedních pixelů nenulových, přiřadí se danému pixelu hodnota náhodně vybraného souseda. Pokud dojde k situaci, že okolní sousedi nemají stejnou hodnotu, je potřeba si tuto kolizi zaznamenat do pomocné tabulky.

V druhém kroku algoritmu se již prochází obrázek, který má obarveny všechny oblasti, některé však mají více barev díky kolizím. Tyto kolize se opravují právě v tomto druhém kroku pomocí záznamů v pomocné tabulce. V této chvíli již každé oblasti odpovídá jedna hodnota (barva či štítek).

3 Návrh a implementace detektoru

Po nastudování metod zaměřených na detekci byl navrhnut detektor objektů, jehož schéma je na obrázku 3.1. Jako implementační jazyk byl v souladu se zadáním diplomové práce zvolen jazyk C++ s podpůrnou knihovnou pro zpracování obrazů OpenCV. Implementované funkční bloky byly naprogramovány jako samostatné knihovny a jsou spojovány do funkční posloupnosti voláním těchto knihovních funkcí a využitím jejich tříd.

Při řešení práce byl kladen důraz především na maximálním využití již vytvořeného kódu z knihovny OpenCV, která je specializovanou knihovnou pro zpracování obrazu. Můžeme tedy předpokládat, že implementace daných algoritmů je v této knihovně odladěny a optimalizovány. Další výhodou této knihovny je její multiplatformnost, jež byla rovněž vyžadována v zadání. Aby bylo dosaženo maximální možné přenositelnosti, a to i na úrovni zdrojových kódů, bylo pro zpracování projektu vybráno vývojové prostředí Eclipse, jež je portováno na drtivou většinu současných platforem.

Jak už bylo řečeno dalším požadavkem na detektor byla rychlost výpočtu. Využil jsem tedy výhody dosti ideálních podmínek pro detekci ve zvoleném prostředí a navrhl jsem co do funkčnosti plnohodnotný detektor s minimálními nároky na výpočetní čas.

3.1 Detekce objektů

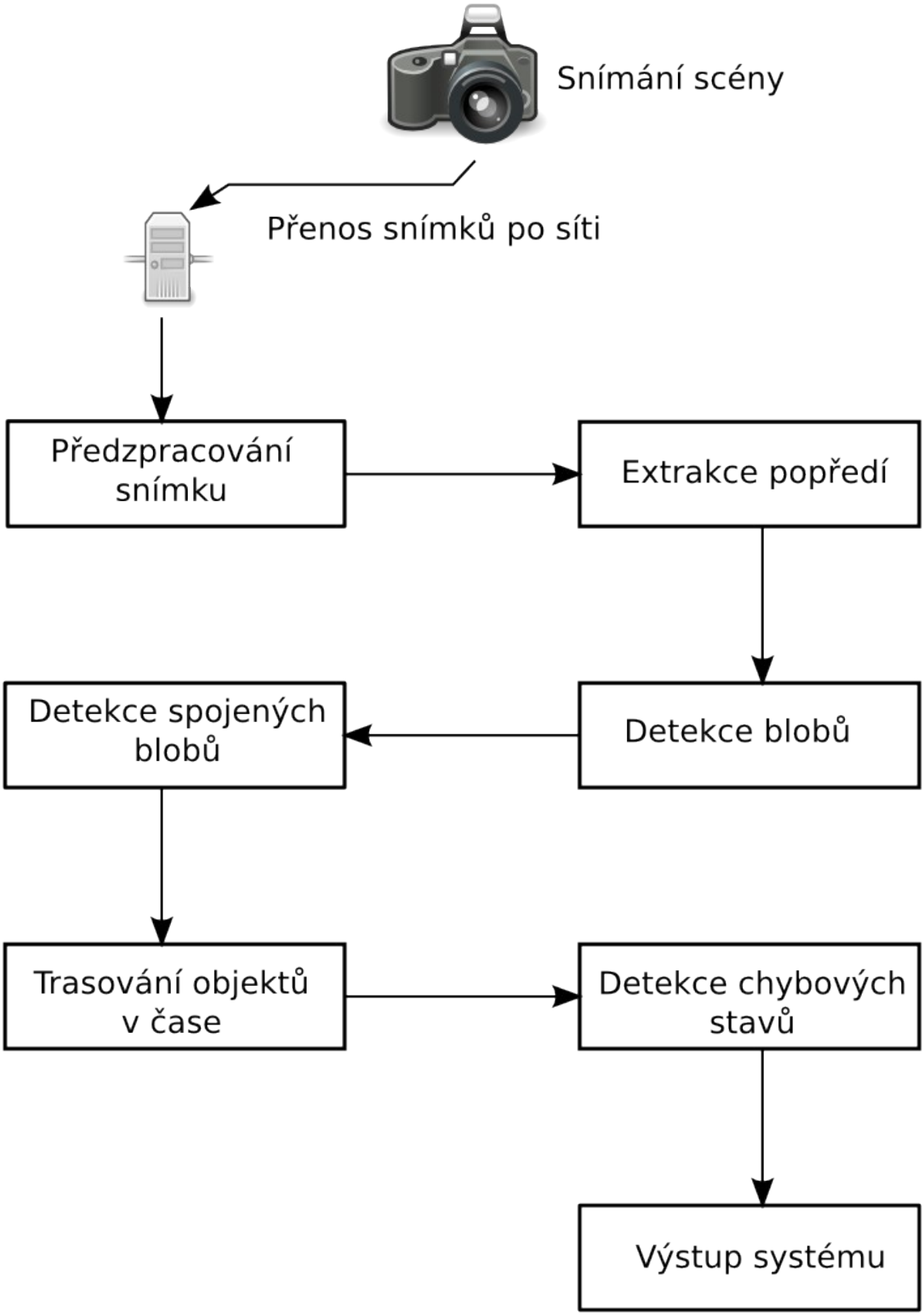
Detekci objektů lze obecně definovat jako rozpoznávání statických i pohyblivých objektů v obraze a jejich následné sledování v prostoru i čase. V našem konkrétním případě jde o detekci objektů na dopravníkovém páse, lze tedy úspěšně předpokládat, že tyto objekty jsou pouze pohyblivé a dokonce s konstantním rychlostním a směrovým pohybovým přírůstkem. Tato skutečnost plyne z optimálních podmínek umístění kamery, která je napevno umístěna nad pásem a snímá jednotlivé objekty z profilu, a také z konstantně se pohybujícího pásu.

Návrh a implementace celého systému se tímto faktem velmi zjednodušuje, především pak blok zabývající se trasováním objektů. Dalším faktorem, jenž zjednodušuje situaci oproti obecnému sledování objektů, je fakt, že sledované objekty se během detekce pohybují po rovině, jež je rovnoběžná s rovinou, kterou snímá kamera a nemusí se tedy pracovat s perspektivou v 3D prostoru.

V celém systému jde hlavně o volbu vhodného poměru kvality detekce ku rychlosti zpracování. Celý systém musí pracovat v reálném čase a odchylka výsledků detekce musí být samozřejmě co nejmenší, s určitou chybovostí se však musí počítat vždy.

Po seznámení se a po prostudování problematiky průmyslových sledovacích aplikací jsem navrhl schéma systému, jenž odráží všechny aspekty navrhovaného řešení. Toto schéma je vyobrazeno na obrázku 3.1 a v následujících odrážkách jsou popsány jednotlivé bloky tohoto schématu.

- Vstupem systému je senzor, v úlohách pro detekci objektů bude úlohu senzoru většinou zastávat videokamera, která bude snímat dopravníkový pás. Informace jdoucí ze senzoru budou mít v našem případě podobu video streamů nebo sekvencí snímků.
- Samotný výpočet detekce probíhá na počítači, jenž je od kamery fyzicky oddělen přes rozhraní. V průmyslových podmínkách, kde nelze zaručit bezprašnost prostředí, popřípadě není možné umístit počítač do blízkosti kamery, jsou informace ze senzoru zasílány po lokální počítačové síti na počítač, kde běží aplikace zpracování obrazu. V případě, že je kamera umístěna v blízkosti počítače, je možné využít jiné rozhraní pro přenos snímků, např. FireWire nebo USB. Tato rozhraní přinášejí obecně lepší výkon než využití síťového přenosu.
- V modulu předzpracování probíhá výběr relevantní oblasti detekce, úprava barevného prostoru, filtrace a ekvalizace obrazu. Tento modul je obecně možné provádět již v samotném senzoru, pokud to senzor umožňuje. Zoptimalizuje se pak čas potřebný pro přenos informací (snímků) po přenosovém kanále.
- Modul extrakce popředí se stará o oddělení popředí od pozadí obrazu. Výstupem z modulu je maska popředí, se kterou se následně pracuje. Maskou popředí se rozumí binární obraz, jenž bílou barvou představuje popředí a černou pozadí.
- V následujícím modulu detekce blobů se provádí segmentace obrazu na jednotlivé bloby. V konkrétním snímku jsou nalezeny spojité oblasti, které se prohlásí za nadetekované bloby. Soubor informací o těchto blobech je vstupem do modulu trasování objektů v čase.
- Modul detekce spojených blobů se stará o detekci a opravu situací, kdy dva blízko umístěné objekty jsou považovány za jeden velký blob.
- V modulu trasování objektů se sledují jednotlivé bloby, které jsou v této chvíli již považovány za objekty, v temporální oblasti. Na základě těchto informací se určí jejich trajektorie a predikuje se jejich pozice na následujícím snímku.
- Detekce a oprava chybových stavů se stará o ošetření nestandardních situací např. nežádoucí objekt překrývající objekty, výpadek sítě atd.
- Výstupní modul se stará o prezentaci výsledku detekce uložením patřičného údaje do databáze nebo do lokálního logovacího souboru.



Obrázek 3.1: Schéma detektoru

Po získání testovacích dat byla provedena prvotní implementace systému, jež byla otestována na těchto datech, která byla pořízena během implementace testovací verze softwaru. Po této fázi vývoje byla zakoupena a umístěna kamera nad dopravníkový pás v provozu firmy Techservis Láník Boskovice.

Zpracování obrazu samotného by se dalo popsat pomocí následujícího pseudokódu.

```
stream=InitNetStream();          //inicializace streamovaného zdroje
objects=new Set of Object;       //inicializace prázdné množiny nalezených
objektů
while(image=stream.getImage()){       //Načítání jednotlivých obrázků
        roi=getSubImage(image);       //Vyjmuti relevantního regionu
        grayImage=convertRGB2GRAY(roi);              //Převod na odstíny šedi
        grayImage=correctLensDistortion(grayImage);//Korekce zakřiveni
                                                     //čočky
        mask=threshold(grayImage);    //Vyprahování
        mask=dilate(mask);            //Zaplnění děr
        blobs=findBlob(mask);         //Nalezeni blobů
        removeSmallBlob(blobs);       //Odstranění velmi malých blobů
        foreach(blob from blobs){     //Iterace přes všechny nalezené bloby
                if(blob.size>size){   //Pokud je blob příliš veliký
                        checkSplitBlob(blob,blobs);  //Zkontroluje se jestli
                                                     //nejde o spojený blob
                }
        }
        foreach(blob from blobs){     //Iterace přes všechny nalezené bloby
                updateTraceObject(blob,objets)       //trasovani objektu
        }
        rempoveOldObject(objets);     //Odstranění starých objektů
}
```

3.2 Senzor a předzpracování obrazu

Tato kapitola je zaměřena na popis problematiky výběru vstupního senzoru a jeho fyzické realizace nad dopravníkovým pásem. Dále je zde rozebrána problematika předzpracování obrazu jakožto výstupu senzoru a jeho optimalizace pro další zpracování.

3.2.1 Výběr kamery a její umístění

Výběr samotného snímacího senzoru, v našem případě kamery, je velmi citlivá část celé implementace systému. I výborně navržený detektor objektů může dávat podprůměrné výsledky, pokud má nekvalitní vstupy. Proto je potřeba výběru vhodné kamery věnovat patřičnou pozornost.

Pro detekci objektů na dopravníkovém páse musí kamera dodávat dostačující počet snímků za sekundu (dále fps - frames per second). Výběr minimální vhodné hodnoty fps nejvíce ovlivňuje rychlost dopravníkového pásu, kde by při malém fps a rychlém pásu mohlo docházet k chybám detekce.

Dále by měla mít kamera dostačující rozlišení obrazu pro danou detekční úlohu. Tento parametr je velmi specifický na daném problému, v našem případě se jako dostačující rozlišení jeví 640*480 pixelů.

Dalším významným faktorem, který je potřeba vzít v úvahu při návrhu, je způsob jakým kamera komprimuje snímky a zasílá dále ke zpracování. Kamery využívají vesměs dva způsoby zasílání snímků, a to jednak zakódováním do video streamu, nebo zasíláním jednotlivých snímků na vyžádání. První uvedený způsob je efektivnější, protože lze dosáhnout větší komprese dat.

Fyzické připojení senzoru k systému je další faktor, který velmi ovlivňuje výběr kamery. Jako vhodný způsob připojení se jeví FireWire, popřípadě využití lokální sítě (ethernet). Jako nevyhovující způsob považuji připojení skrze bezdrátovou síť WiFi, kde by mohlo docházet k nepředvídatelným časovým prodlevám a nespolehlivosti spojení.

Po analýze detekční úlohy ve výrobě firmy Techservis jsem vybral kameru D-Link DCS-900, jejíž tabulkové hodnoty se jevily jako vhodné. Tyto tabulkové hodnoty jsou uvedeny v příloze Dodatek C.

Tato kamera byla umístěna nad výrobní linku na speciální kovové kostře, která byla pro tyto účely zkonstruována. Následně byla kamera připojena k počítačové síti skrze přivedený UTP kabel. Z etických důvodů byla kamera nastavena tak, aby snímala pouze přesně vyhraněný úsek dopravníkového pásu a nesnímala obsluhu pásu. Obrázek s detailem upevnění kamery nad linkou je v příloze Dodatek F.

Pro testovací účely bylo nezbytné, aby byla kamera dosažitelná i z vnější sítě (z internetu). Z bezpečnostních důvodů nebylo možné nastavit této kameře veřejnou IP adresu, nicméně byl problém vyřešen pomocí přesměrování portu 3743 ve firemním směrovači na vnitřní IP adresu a port 8080.

Po namontování kamery se objevil problém s vysokou mírou vodních par vycházejících z kontinuální mikrovlnné pece, jež je součástí dopravníkového pásu, a detekované objekty vstupují těsně za touto pecí do detekční zóny. Z těchto důvodů jsem se obával, že dojde k orosení čočky kamery, což by mohlo mít neblahé následky jednak pro samotnou detekci objektů a jednak by mohlo dojít k poškození kamery samotné. Naštěstí se tato obava ukázala jako planá, mimo jiné také díky tomu, že kamera byla nakonec umístěna výše než bylo původně předpokládáno. V této výšce jsou již vodní páry natolik rozptýlené, že nezpůsobují žádné problémy.

Bohužel byl u této kamery zjištěn podstatnější problém, a to, že není schopna do vzdálené sítě (skrze internet) dodávat více jak 3 fps, což znemožnilo důkladnější otestování. Při připojení po vnitřní síti se fps vyšplhalo na 6 fps, což už se jevilo jako dostatečná vzorkovací frekvence.

Kamera DCS-900 obsahuje vestavěný webový server, skrze který se provádí nastavení kamery. Nastavení je rozdělené do samostatných podsekcí, ze kterých jsou nejvíce zajímavé sekce zabývající se nastavením snímání scény a nastavením síťových parametrů.

Kamera produkuje snímky ve formátu jpeg s nastavitelnou 5-úrovňovou kompresí, tato komprese byla nastavena na minimální, aby nedocházelo ke zkreslení snímku. Dále byla nastavena snímkovací frekvence na 50Hz, z důvodu synchronizace s okolním osvětlením a rozlišení na 640*480 pixelů.

V subsekci zabývající se síťovým nastavením byly zvoleny následující parametry.

- Naslouchací port 8080.
- Lokální IP adresa, jež byla přidělena správcem sítě.

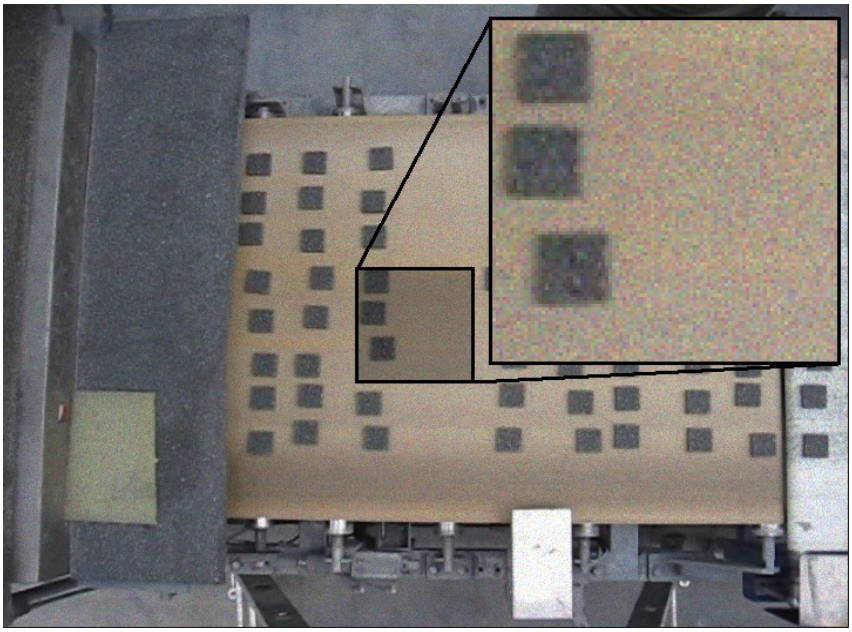
Pro získání snímku přes síť byla navržena knihovna NetImageLoader, která je naprosto nezávislá na zbytku implementovaných tříd a je ji možno obecně použít v kombinaci s knihovnami OpenCV a curl. Více o knihovně NetImageLoader v kapitole 3.5.

3.2.2 Předzpracování obrazu

Vstupní senzor, v našem případě kamera, je zatížen šumem. Jedná se o takzvaný barevný šum produkovaný přímo senzorem. K tomuto šumu dochází jednak při značném zesílení slabého signálu, tedy při zvýšení citlivosti senzoru, a také při expozicích s krátkým časem.

Dá se tedy obecně říci, že šum je závislý na velikosti snímače. Kdy na větší snímač dopadá více světla, což má za následek lepší signál a tím pádem je i přesnější informace o barvě.

Z uvedeného vyplývá, že barevným šumem jsou více zatíženy snímače s menšími snímacími senzory (např. malé kompaktní fotoaparáty).



Obrázek 3.2: Demonstrační obrázek zatížený barevným šumem a detailní výřez

V detailu obrázku 3.2 si můžeme všimnout nesourodosti pozadí pásu, kde jednotlivé pixely tohoto výřezu, mají značně odlišné odstíny. Tento typ šumu lze s úspěchem odstranit pomocí mediánového filtru, popřípadě je možno použít některý z konvolučních filtrů. Více o filtraci v kapitole 2.4.

3.2.3 Oprava geometrie

Při snímání scény kamerou se často vyskytuje porušená geometrie obrazu z důvodu nedokonalosti optické soustavy kamery. Tyto nepřesnosti se dají velmi snadno eliminovat mimo jiné také, protože jsou po celou dobu zpracování obrazu (detekce objektů) neměnné.

Obrázek 3.3: Vyznačená místa s viditelným zakřivením

V naprosté většině případů stačí při instalaci zařízení provést kalibraci systému pomocí snímání kalibračních obrazců, ze kterých lze následně určit korekční parametry. Většinou se jedná o určení projekce čočky, která způsobuje vypuklou deformaci obrazu (lens distortion).

Na obrázku 3.3 je snímán dopravníkový pás, na okrajích obrázku si můžeme povšimnout typické optické vady snímacího zařízení tzv. rybího oka. Inverzní transformace k zakřivení čočky se dá vypočítat následující rovnicí 3.1.

$$P'_x = P_x(1 - a_x \|P\|^2) \qquad P'_y = P_y(1 - a_y \|P\|^2) \tag{3.1}$$

Kde P je vektor souřadnic bodu, $\|P\|$ je modul vektoru a A je projekce čočky. Pro většinu objektivů platí, že a_x a a_y jsou stejné hodnoty. Pro účely korekce je však důležitější reverzní funkce.

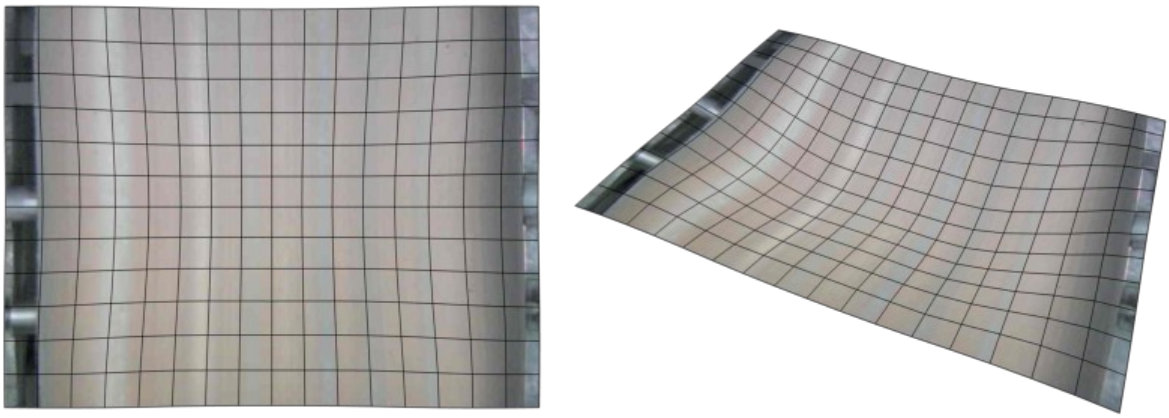
$$P_x = \frac{P'_x}{1 - ax\left\|\frac{P}{1 - ax\|P\|^2}\right\|^2} \qquad P_y = \frac{P'_x}{1 - ay\left\|\frac{P}{1 - ay\|P\|^2}\right\|^2} \tag{3.2}$$

Takto vypočítané transformované souřadnice jsou normalizovány na rozsah -1 až 1, je proto potřebné je přepočítat zpět na souřadnice obrázku.

$$i = \frac{(P_x + 1) * w}{2} \qquad j = \frac{(P_y + 1) * h}{2} \tag{3.3}$$

Kde w je šířka a h šířka obrazu, i je souřadnice v rozmezí 0 až (w-1) a j souřadnice v rozsahu 0 až (h-1).

Na obrázku 3.4 je znázorněna transformace souřadného systému. Na obrázku vlevo je vyobrazen opravený obrázek se sítí, na které je vidět průběh zhroucení sítě při korekci. Na obrázku vpravo je pak znázorněna tato deformace ve 3D prostoru.



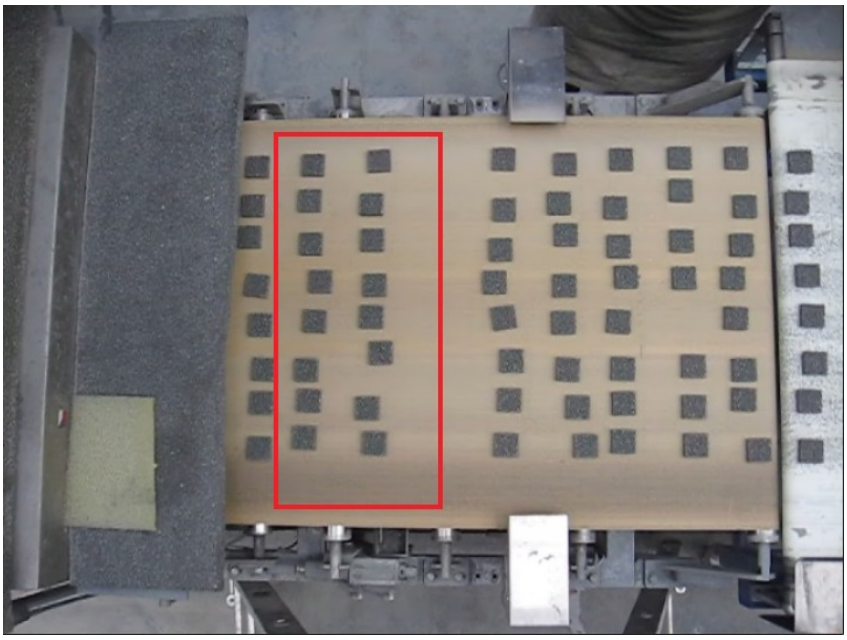
Obrázek 3.4: Ilustrační obrázek transformovaného obrazu

V rámci diplomové práce byla tato funkce naimplementována a je uvedena v příloze Dodatek A . Pro kalibraci kamery a zjištění zakřivení čočky byl navržen kalibrační obrazec taktéž obsažen v příloze Dodatek B. Na obrázku uvedeném v příloze Dodatek D je vyobrazen výsledek výše uvedené funkce na testovací obrázek lena.jpg s různými parametry zakřivení.

3.2.4 Oblast detekce

Pro efektivní řešení daného problému, v našem případě detekce objektu na dopravníkovém páse, není vhodné provádět většinu operací nad celou oblastí obrazu. Na následujícím obrázku 3.5 je znázorněna relevantní oblast pro detekci objektů též označována ROI (region of interest). Zbylé části obrázku, mimo tuto oblast, jsou jednak pro detekci nepodstatné a jednak v nich nelze zaručit optimální podmínky pro detekci (např. oblast, kde se pohybují další objekty).

Navíc pokud se tímto způsobem vymezí oblast, jež nepokrývá celý vstupní obraz, je možné provádět dodatečnou korekci třasu, která je zejména v průmyslových aplikacích všudypřítomná a jež může být potenciálně nepříjemným zdrojem detekčním chyb.



Obrázek 3.5: Výběr relevantní oblasti detekce objektů

Na obrázku 3.5 je obdélníkem naznačena potenciálně výhodná oblast pro další zpracování při detekci objektů na páse. Na obrázku si lze také všimnout špatně umístěné kamery, která snímá příliš mnoho zbytečného prostoru na úkor efektivní oblasti pro detekci, jež má proto zbytečně malé rozlišení.

3.3 Detekce blobů a jejich trasování

Z výstupu bloku pro extrakci popředí vychází maska popředí, ve které je potřeba nadetekovat jednotlivé objekty. Objektem se v masce rozumí souvislá oblast pixelů masky, kde souvislost je chápana jako topologická sousednost jednotlivých pixelů. Detekcí je zde míněna segmentace obrazu, objekty jsou tedy nalezeny segmentací masky. Těmto objektům budeme říkat bloby.

Maska je segmentována pomocí algoritmu connected component labeling (déle jen CCL) který je popsán v kapitole 2.7.3. V této kapitole je rozebráno více segmentačních metod, nicméně právě CCL, byla vybrána jako vhodná metoda pro danou úlohu, jednak díky dostačujícím výsledkům a mimo jiné také pro svoji výpočetní nenáročnost.

Tato segmentace je prováděna s poměrně velkou úspěšností. Jediným případem, kdy dojde k chybě detekce, je situace, kdy se na jedoucím páse ocitnou objekty v přílišné blízkosti, popřípadě když se do oblasti snímání dostane ruka obsluhy pásu. Tyto nepříjemné situace a jejich řešení jsou rozebrány v kapitole 3.4 . V následující subkapitole je popsán jednoduchý trasovací algoritmus, který byl v rámci projektu navržen.

3.3.1 Trasování objektů

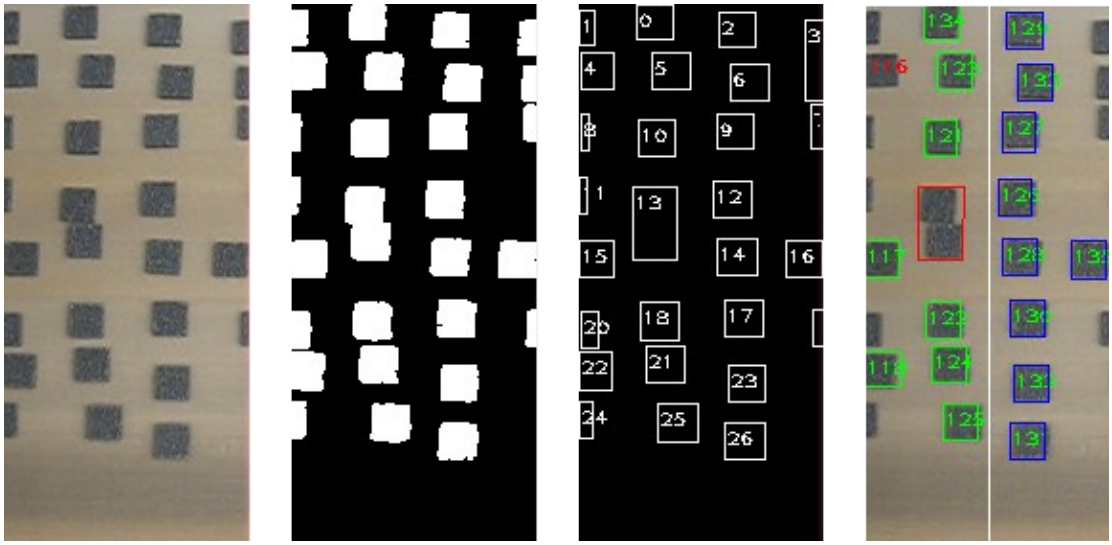
Jak bylo výše popsáno v každém snímku jsou nalezeny bloby reprezentující objekty v daném časovém okamžiku. Vyvstává tedy potřeba sledovat tyto objekty v čase, dále predikovat jejich pohyb a určit zda daný je totožný s objektem z předchozího snímku, popřípadě že se jedná o objekt nový.
Vzhledem k faktu, že se jedná o trasování objektů na jedoucím páse, je tento krok značně zjednodušen skutečností, že objekty se pohybují konstantní rychlostí a stále stejným směrem. Tento fakt vyplývá z rovnoměrně se pohybujícího dopravníkového pásu a staticky umístěné kamery.

Obecně je však tento problém složitější a vyžaduje robustnějšího zpracování, např. v aplikacích pro sledování dopravní situace, popřípadě v systémech pro sledování lidí se musí trasovací algoritmus vyrovnat také například s rotací objektů, perspektivou scény, chaoticky se pohybujícími objekty atd.

Vzhledem k účelu, ke kterému má trasovací algoritmus sloužit, jsem návrh jednoduchý princip optimalizovaný na daný problém a časovou náročnost výpočtu. Pro zápis algoritmu jsem zvolil pseudojazyk podobný jazyku C++, ve kterém je celý systém naimplementován.

```
Global objects=new Set of Object();     //inicializace prázdné množiny  nalezených objektů
....
updateTraceObject(blob,objets){
    foreach(blob from blobs){                       //Iterace přese všechny nalezené bloby
        foreach(object from objects){               //Iterace přese všechny trasované objekty
            if(object.isNewPosition(blob)){         //kontrola zda je blob potenciální posunuty
                                                    //objekt
                object.updatePostion(blob);         //nastaveni nové pozice objektu a upravení
                                                    //vektoru posunu
                foreach break;                      //ukonceni iterace přeze objekty
            }
        }
        blobs.put(new Object);          //pokud nebyl blob přiřazen žádnému objektu přidá
                                        //se do seznamu jako nový objekt
    }
    foreach(object from objects){       //Iterace přese všechny trasované objekty
        if(!object.isDetect)            //pokud nebyl objekt v současném snímku
                                        //nadetekován
            object.hideCount--;         //zvýší se počítadlo kdy byl object nezdetekovan
            if(object.hideCount>MAX_HIDE_COUNT){   //pokud toto číslo překročí mez
                objects.remove(object);            //je objekt vymazán
            }else{
                object.move();          //posunuti o posledni známý vektor
            }
        }
    }
}
```

Na obrázku 3.6 je zobrazen postup zpracování obrazu z hlediska výstupu jednotlivých funkčních bloků popsaných v kapitole 3.1. Na prvním obrázku zleva je originální obraz, na dalším snímku je zobrazena maska popředí, dále pak následuje ilustrace nadetekovaných blobů z této masky. Na posledním obrázku je zobrazen výsledek trasování a počítání objektů, kde je modrým rámečkem označen objekt, který ještě nebyl započítán, zeleným pak již započítaný objekt a červeným příliš velký blob. Bílá čára uprostřed je dělící čárou, kdy projetím objektu přes tuto čáru dojde k jeho započítání.



Obrázek 3.6: Ilustrace zpracování detekce objektů

3.3.2 Počítání objektů

Základní funkčností cílového systému má být výpočet projetých objektů po dopravníkovém páse za určitý časový úsek. Započítaným objektem se rozumí, jak již bylo řečeno v komentáři k obrázku 3.6 , objekt, který projel skrze dělící čáru.

Tato definice není úplná, může totiž dojít k situaci, kdy byl pás zastíněn cizím objektem např. rukou obsluhy pásu (více v kapitole 3.4.1)a daný objekt nebyl při průjezdu přes dělící čáru vidět. Je proto nutné provádět při každé iteraci nad snímky kontrolu, zda všechny objekty, které se nachází v oblasti za dělící čárou, jsou již započítány.

Tento jednoduchý princip vyžaduje pouze jediný nastavovací parametr a tím je pozice této dělící čáry, která byla v implementaci umístěna zhruba doprostřed snímaného prostoru viz. obrázek 3.6.

Algoritmus počítání by se dal popsat pomocí pseudokódu následovně.

```
DetectionLine=200;                  //X pozice dělící čáry
counterObject=0;                    //Globální počítadlo nadetekovaných objektů
...
countObject(objects){
    foreach(object from objects){   //Iterace přese všechny trasované objekty
        if(!object.isCount)         //pokud nebyl objekt ještě započítán
            object.isCount=true;    //Objekt se označí jako započítaný
            counterObject++;        //Zvýší se počítadlo objektů
        }
    }
}
```

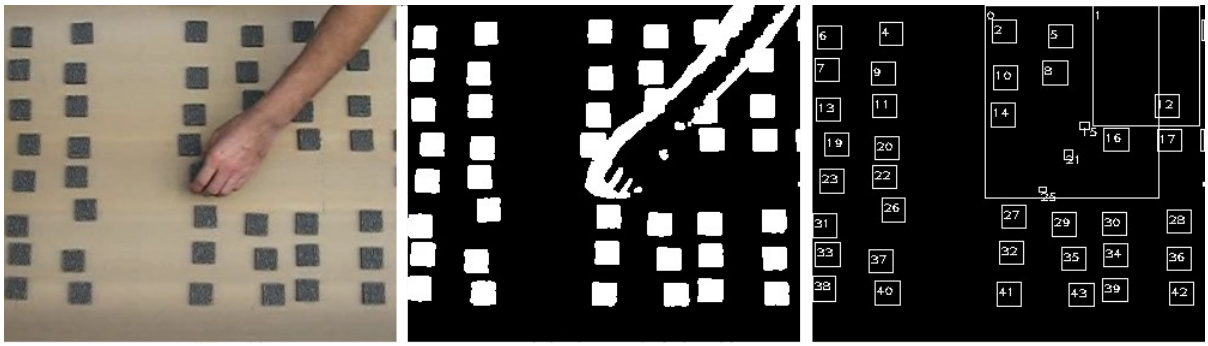
3.4 Ošetření chybné detekce

Během zpracování snímků dochází přirozeně k situacím, ve kterých by systém bez tohoto ošetření nefungoval korektně. Tyto situace nastávají především při těsné blízkosti jednotlivých objektu na páse, kde jsou tyto objekty detekovány jako jeden velký objekt. Další situací, ve které typicky vzniká chybná detekce, je, když se nad pásem ocitne ruka popřípadě jiný nežádoucí objekt, jenž svým tělem překrývá pás a objekty nejsou přes něj vidět.

3.4.1 Cizí objekt ve scéně

V aplikacích pro rozpoznávání objektů na jedoucích dopravníkových pásek nelze vždy zaručit ideální podmínky snímání, proto v těchto případech může docházet k nechtěnému vniknutí cizího objektu do scény. Jedná se především o ruce operátorů u pásu. Aplikace pro rozpoznávání objektů by měla být oproti takovýmto situacím ošetřena.

Při vniknutí cizího objektu do scény se většinou překryjí již detekované objekty tímto nechtěným tělesem, což má za následek fiktivní zmizení již detekovaných objektů ze scény, popřípadě dojde k překrytí právě přijíždějících objektů, ty se pak po znovuobjevení mohou opětovně započítat, čímž dojde k chybě. Aby se předešlo těmto chybovým situacím, je potřeba aplikaci vybavit mechanizmem, jenž dokáže tyto situace detekovat a efektivně předejít.

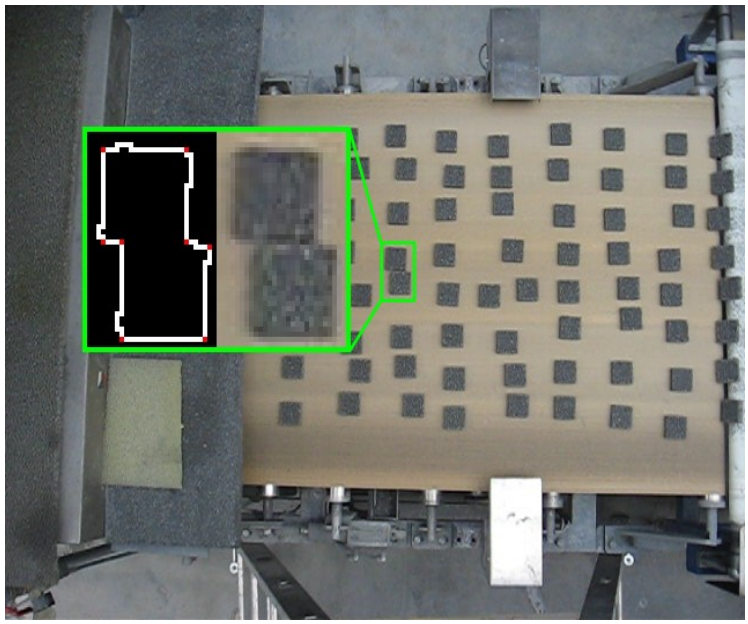


Obrázek 3.7: Zprava originální obraz, vyprahovaný obraz, obálky nadetekovaných blobů

Tento mechanizmus je již částečně obsažen v trasování objektů v temporální oblasti. A to tak, že pokud objekt zmizí, je ještě nějaký čas predikován jeho potencionální pohyb. Tímto je vyřešen problém s vícenásobným započítáním stejného objektu . Vyvstává ale potřeba detekovat cizí objekt, aby nedošlo k jeho započítání. U naprosté většiny případů je cizí objekt ve scéně obsažen jen částečně, a to mimo jiné znamená, že nadetekovaný blob reprezentující tento objekt přímo hraničí s okrajem detekční oblasti, lze jej proto snadno zjistit.

3.4.2 Spojené bloby

Situace, kdy se na páse ocitnou objekty v těsné blízkosti je celkem běžná. Robustní ošetření této situace je tedy nezbytným krokem, a to nejen z hlediska detekce této situace ale i jejího korektního vyřešení. Pokud by tato situace nebyla žádným způsobem řešena, byl by vzhledem k četnosti situace počet nadetekovaných objektů mnohem menší, než by odpovídalo skutečnosti.



Obrázek 3.8: Ukázka spojených objektů. V detailu vyextrahované hrany s vyznačenými rohy

Na obrázku 3.8 je znázorněna situace z testovacích dat, kdy k takové kolizi došlo. Detekce této situace je poměrně jednoduchá, zjednodušeně řečeno, detekce vychází z nadměrné velikosti tohoto blobu. Samotné vyřešení (rozpoznání, že jde o dva bloby v těsné blízkosti) je o poznání složitější. V prvním kroku rozpoznání spojených blobů se na vyprahovaném a zdilatovaném detailu toho velkého multiblobu vyhledají rohy pomocí Harrisova detektoru a následně vyhledají lokální maxima. V této chvíli tedy dostáváme masku bodů, kde body reprezentují rohy, bohužel v daném rozlišení se tato detekce jeví jako značně nespolehlivá, tento handicap je však eliminován schopností trasovacího algoritmu vyrovnávat se s občasnou ztrátou blobu. Z masky rohů se vyextrahují jednotlivé bloby následovně.

Z analýzy problému vyplývá, že detekované objekty jsou vždy čtvercového profilu, z toho plyne, že vzdálenost mezi sousedními rohy je dána následujícím vztahem.

$$d_1 = a \quad d_2 = \sqrt{a^2 + a^2} \tag{3.4}$$

Kde d_1 je vzdálenost mezi sousedními rohy (sousednost po hraně), d_2 je vzdálenost bodů po úhlopříčce. V daném rozlišení se dá říci $d_1 \approx d_2$.

Za korektní blob je tedy považována množina rohu R pro kterou platí.

$$R=\{x \mid x \in M, \forall r \in R \wedge r \neq x \wedge d_1 \leq |xr| \leq d_2\} \tag{3.5}$$

Kde R je výsledná množina bodů, x je roh z celkové množiny všech nalezených rohů M, d_1 a d_2 mají stejný význam jako v předchozím případě a |xr| je vzdálenost mezi rohy x a r. Jinými slovy je R množina takových bodů, pro které platí, že jejich vzájemná vzdálenost je větší nebo rovna než d_1 a menší nebo rovna jak d_2.

V případě nalezení takovéto množiny je do seznamu nalezených blobů přidán další blob s odpovídajícími parametry.

3.4.3 Výpadek sítě

V rámci podnikové sítě by výpadky počítačové sítě neměly být častým jevem, nicméně i k takovým situacím může dojít, a sledovací systém by na takové případy měl být připraven. V naprosté většině situací je výpadek počítačové sítě doplněn i výpadkem elektřiny a vzhledem k tomu, že výrobní linka i počítačová síť je napojena na stejný záložní obvod, je proto vyloučeno, že by došlo k nezapočítání některých objektů.

V případě, kdy vypadne pouze počítačová síť, je o incidentu vytvořen záznam v databázi, popřípadě v logovacím souboru. A později může být záznam o incidentu vyřešen.

3.5 Uživatelský manuál

Pro úspěšné spuštění programu je vyžadováno, aby na cílovém počítači byly nainstalovány knihovny OpenCV a curl.

Parametry spuštění se nastavují přímo na příkazové řádce a pomocí konfiguračního souboru. V následujících odrážkách jsou popsány přepínače příkazové řádky

- -h při zadání tohoto přepínače vypíše program na standardní výstup návod pro použití programu
- -config <cesta k souboru s konfigurací> přepínač říká programu, že má použít soubor s konfigurací a informuje ho, kde má tento soubor hledat

Konfigurační soubor má následující formát, na každém řádku je definována vždy maximálně jedna vlastnost s n parametry. Za názvem vlastnosti je vyžadován znak =, za tímto znakem následuje n parametrů této vlastnosti. V konfiguračním souboru jsou definovány následující vlastnosti, přičemž nezáleží na pořadí jednotlivých řádků.

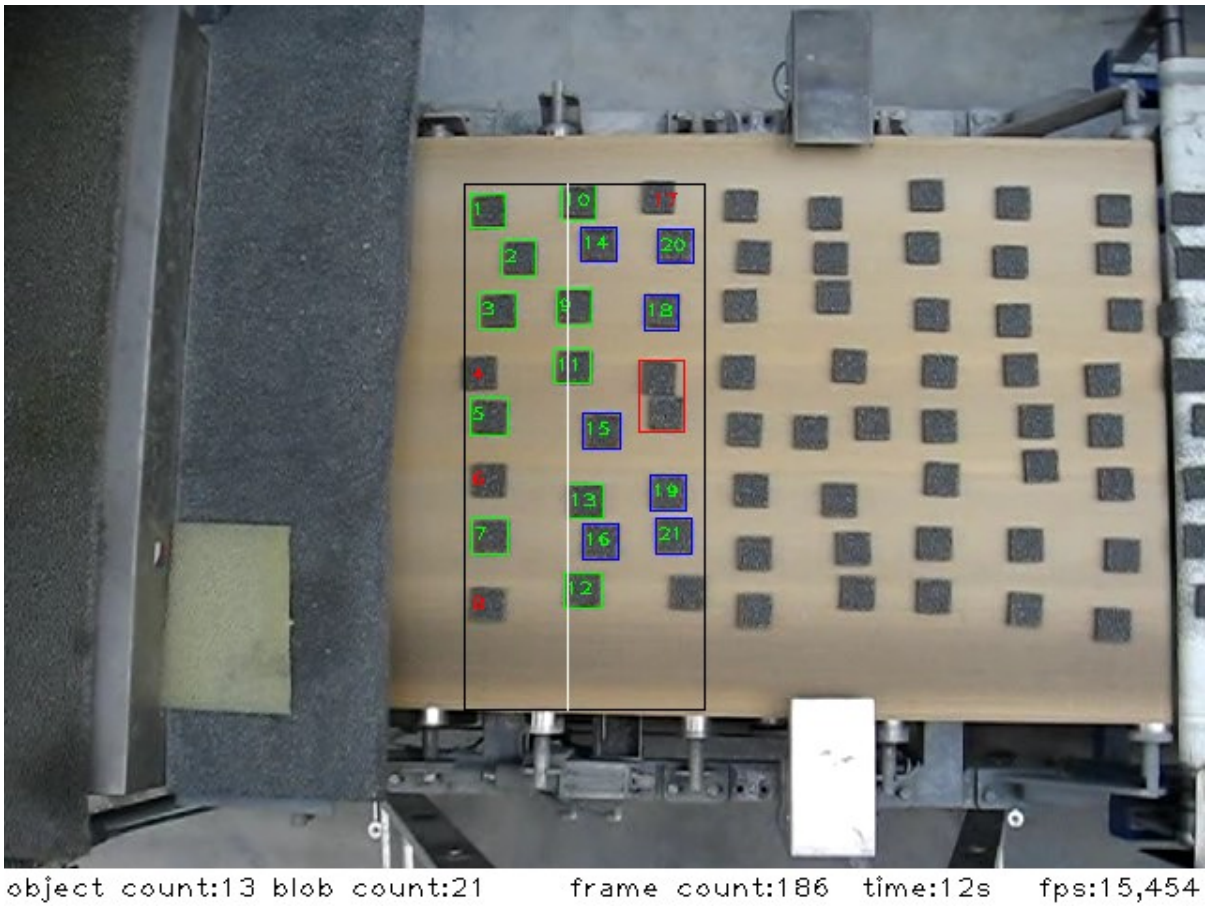
- source <string> zdroj snímků
- output <string> vlastnost nastavuje výstupní soubor
- window_size <int[2]> rozměr detekčního okna
- window_position <int[2]> pozice detekčního okna
- detection_line_pos <int> x-pozice detekční linie
- gui <boolean> vlastnost udává zda se má vykreslovat gui
- dx <double> vlastnost udává povolenou odchylku pohybu objektu mezi jednotlivými snímky v x-ose
- dy <double> vlastnost udává povolenou odchylku pohybu objektu mezi jednotlivými snímky v y-ose
- object_size <float> vlastnost udává délku hrany čtverce představující půdorys objektu. Délka je zadávána v metrech
- ratio <float> vlastnost představuje měřítko. Jedná se vlastně o poměr počtu metrů na jeden pixel

Ukázka konfiguračního souboru je uvedena v Dodatek E.

3.6 GUI

Na následujícím obrázku 3.9 je ukázáno GUI (graphics user interface). Černý obdélník označuje oblast detekce. Modrými obdélníky jsou zobrazeny obálky ještě nezapočtených objektů, zelenými pak obálky započtených objektů. Červeným obdélníkem jsou označeny ty bloby, které jsou detekovány jako špatné objekty. Každý trasovaný objekt při vstupu do detekčního obdélníku získá jedinečné identifikační číslo (dále jen ID), které se vykresluje uprostřed detekovaného objektu. Pokud toto číslo zčervená, znamená to, že pro daný objekt nebyl v daném snímku nalezen blob. Toto se typicky stává při zastínění pásu rukou a v situaci, kdy objekt vyjíždí z oblasti detekce.

Ve spodní části okna je informační panel s dodatečnými informacemi. Zleva se jedná konkrétně o tyto údaje, počet započítaných objektů, počet nadetekovaných blobů v aktuálním snímku, počet zpracovaných snímků, čas zpracování a počet snímku, které se zpracují za jednu sekundu.



Obrázek 3.9: GUI programu

Předčasné ukončení vykonávání rozpoznání je umožněno pomocí stisknutí klávesy ESC na klávesnici.

4 Výsledky

Program byl odladěn na testovacích datech, na kterých měl 100% úspěšnost. V reálném provozu se program bohužel kvůli problémům s kamerou, které jsou popsány v kapitole 3.2.1, nepodařilo otestovat.

Nicméně v rámci testovacích dat byly odzkoušeny chybové situace, kdy se na dopravníkovém páse octly spojené objekty a také vniknutí cizího objektu do scény. Tyto situace byly vyřešeny bez větších problémů.

Rychlost zpracování snímků, bez zobrazení testovacího gui, na počítači Lenovo R61 s procesorem Intel Duo Core 2GHz, operační pamětí 2GB a operačním systémem Ubuntu 8.4 dosahovala průměrných hodnot kolem 120 fps. Rychlost zpracování se zapnutým vykreslováním průběhu detekce byla znatelně pomalejší, převážně z důvodu čekání na výsledek rutiny cvWaitKey, která má za úkol načítat uživatelský vstup z klávesnice.

Průměrné vytížení paměti se pohybovalo kolem 8MB, při běhu bez testovacího gui se tato hodnota mírně zmenšila.

Tyto výsledky hodnotím jako velmi příznivé, jelikož pro reálné použití aplikace by stačily mnohem horší hodnoty. Z čehož vyplývá, že aplikace by se dala provozovat na méně výkonné sestavě čímž by došlo ke snížení pořizovacích nákladů na tento systém.

5 Závěr

V této kapitole je proveden souhrn diplomové práce spolu s konfrontací se zadáním práce. První bod zadání, tedy prostudování základů zpracování obrazu a detekce objektů v obraze, byl zpracován v první části technické zprávy. V této části je také rozebrán další bod zadání a to seznámení se s problematikou průmyslových aplikací počítačového vidění.

V následujících kapitolách shrnu obsah této části. Druhá kapitola se zabývá teoretickým zpracováním obrazu pro detekci pohybujících se objektů včetně rozboru jednotlivých fází.

Tyto fáze byly rozčleněny do jednotlivých podkapitol. První podkapitola 2.1 se zabývá rozborem barevných prostorů a jejich přínosu pro zpracování obrazu, včetně jejich využití v konkrétních aplikacích. Následující kapitola 2.2 osvětluje pojem ekvalizace obrazu a jeho místo ve zpracování obrazu v aplikacích zaměřených na detekci objektů. Kapitola 2.3 je zasvěcena morfologickým operacím jakožto možným prvkem předzpracování obrazu před samotnou segmentací. Filtrací obrazu a rozebrání jednotlivých metod se zabývá kapitola 2.4. V kapitole 2.5 jsou nastíněny algoritmy pro extrakci popředí od pozadí jsou zde popsány metody založené na znalosti prostředí (prahování) i zástupce statistických metod (extrakce popředí pomocí GMM).

Následující kapitola 2.6 pojednává o extrakci dalších pomocných příznaků, jenž mohou případnou detekci zkvalitnit jedna se především o detekci hran, detekci rohu a texturní příznaky. V konečném řešení byly použity především metody detekce rohů, pro řešení chybových stavů zejména při rozlišení spojených blobů.

Předposlední kapitolou zabývající se teoretickým rozborem je kapitola 2.7, jenž je zasvěcená segmentaci obrazu na jednotlivé bloby. Zde stojí především za zmínku semínkové vyplňování, které bylo nakonec použito v implementaci systému.

Dalším bodem zadání je výběr vhodné metody a navrhnutí jednoduchého detektoru objektů na pásovém dopravníku. Tento bod je rozebrán v kapitole 3 a pojímá veškeré poznatky a výsledky ze samotné implementace systému. V jednotlivých podkapitolách jsou rozebírány funkční bloky pro detekci objektů na dopravníkovém páse. V kapitole 3.2 je rozebrána problematika výběru vstupního senzoru, jeho parametrů a fyzické umístění. V dalších podkapitolách této kapitoly jsou popsány následné fáze předzpracování obrazu

Samotná detekce blobů a jejich trasování v temporální oblasti je rozebrána v kapitole 3.3. Dále je zde vysvětlen použitý algoritmus počítání detekovaných objektů.

Kapitola 3.4 je zaměřena na popis řešení ošetření chybových stavů. Hlavní pozornost při řešení chybových stavů měla detekce a vyřešení spojení blízkých blobů. V kapitole 3.5 je popsán programový manuál včetně použití příkazové řádky a konfiguračního souboru. Dále se tato kapitola

zabývá podmínkami pro úspěšnou kompilaci programu. Výsledný vzhled aplikace je nastíněn v následující kapitole 3.6.

V rámci bodu zadaní, experimentace s implementací, byla v průběhu posledního semestru provedena montáž kamery nad výrobní linku v provozu firmy Láník Techservis Boskovice. Bohužel vybraná kamera neměla patřičné parametry a nebylo možné sledování online dat skrze vnější síť. Proto odladění systému bylo provedeno na testovacích datech jenž, byly pořízeny během semestrálního projektu.

Poslední dva body zadání tj. programová dokumentace a plakát shrnující diplomovou práci jsou splněné a nachází se uložené v doprovodném CD.

Z původně stanovených cílů nebyl pouze splněn výstupní modul pro prezentaci výsledků do databáze. Další pokračování tohoto projektu tedy vidím v dopracování výstupního modulu a následné dlouhodobé evaluaci v provozu, kde sledovací systém poběží v rámci podnikové sítě, a pozdější sestavení dlouhodobých statistik vytížení výrobní linky a případná optimalizace provozu ze závěru zpracovaného z těchto statistik.

Literatura

[HLA01] prof. ing. Václav Hlaváč CSc, Hledání Hran. Praha, Fakulta elektrotechnická ČVUT 2007.

[HLA02] prof. ing. Václav Hlaváč CSc, Předzpracování v prostoru obrazu. Praha, Fakulta elektrotechnická ČVUT 2007.

[SPA07] Ing. Michal Španěl. Přednáška o statistickém rozpoznávání. ÚPGM VUT Brno. 2007.

[LOR07] Ing. Luboš Lorenc Ph.D. Získávání znalostí z databází. Shlukování. ÚIS FIT VUT Brno. 2007.

[POT07] Ing. Igor Potůček Ph.D.. Přednáška o detekci a parametrizaci objektů v obraze. ÚPGM VUT Brno. 2007.

[DOU07] Ing. Petr Doubek Ph.D., Mean-Shift segmentace, CMP FEL CVUT Praha 2007.

[ZAR04] ŽÁRA, J., BENEŠ, B., FELKEL, P.: Moderní počítačová grafika. ComputerPress, 2004, 448 s., ISBN: 80-7226-049-9.

[JEL07] Ing. Tomáš Jelínek. Diplomová práce.: Detekce pohybujících se objektů ve video sekvencích . VUT Brno 2007.

[WIK01] YUV.:http://cs.wikipedia.org/wiki/YUV . 30.2.2008.

[SBR97] Smith, S., M., Brady, J.m M. A New Approach to Low Level Image Processing. Int. J. of Comp. Vision, 23(1), 45-78, 1997.

[WIK02] Corner detection.http://en.wikipedia.org/wiki/Corner_detection

[HLA03] prof. ing. Václav Hlaváč Csc: Šedotónová matematická morfologie.Fakulta elektrotechnická ČVUT, katedra kybernetiky . Centrum strojového vnímání, Praha 2004.

Seznam obrázků

Seznam příloh

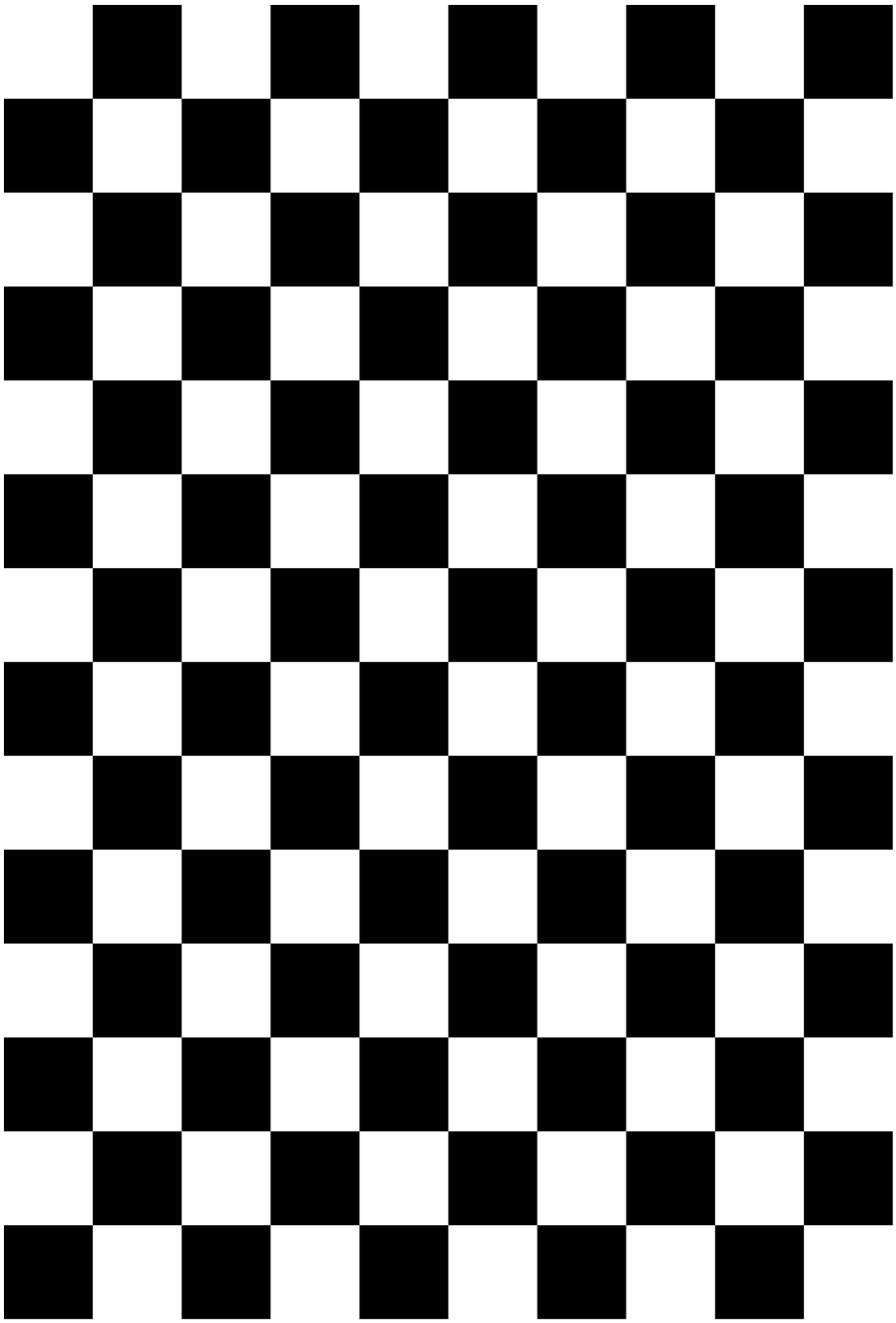
Dodatek A

Implementace korekce zakřivení čočky

```
#include <cv.h>
#include "lensDistortion.h"

/**
 * Funkce provadi opravu zakriveni cocky
 * @param src vstupni obrazek
 * @param dst vystupni obrazek
 * @param aplha zakriveni cocky
 * @return -1 jako chybu v pripade ze nesouhlasi rozmer vstupniho a vystupniho obrazku. 0 pokud vse probehlo
v poradku
 */
int lensDistortion(IplImage* src,IplImage* dst,CvScalar* alpha){
	int w=src->width;
	int h=src->height;
	char* dataSrc=src->imageData;
	char* dataDst=dst->imageData;

	if((w!=dst->width)||(h!=src->height)){
		return -1;
	}
	CvScalar pixel=cvScalarAll(0);
	int i,j,aacount,i2,j2;
	double x,y,x2,y2,r;

	for (j=0; j<h; j++) {
		for (i=0; i<w; i++) {
			aacount = 0;
			pixel.val[0] = 0;
			pixel.val[1] = 0;
			pixel.val[2] = 0;
			x = (2 * i-w) / (double)w;
			y = (2 * j-h) / (double)h;
			r = x*x + y*y;
			x2 = x * (1 - alpha->val[0] * r);
			y2 = y * (1 - alpha->val[1] * r);
			i2 = (x2 + 1) * w / 2;
			j2 = (y2 + 1) * h / 2;

			if (i2 >= 0 && i2 < w && j2 >= 0 && j2 < h) {
				pixel.val[0] += dataSrc[(i2+j2*w)*3];
				pixel.val[1] += dataSrc[(i2+j2*w)*3+1];
				pixel.val[2] += dataSrc[(i2+j2*w)*3+2];
				aacount++;
			}

			if (aacount > 0) {
				dataDst[(i+j*w)*3]=(uchar)pixel.val[0]/aacount;
				dataDst[(i+j*w)*3+1]=(uchar)pixel.val[1]/aacount;
				dataDst[(i+j*w)*3+2]=(uchar)pixel.val[2]/aacount;
			}
		}
	}
	return 0;
}
```

Dodatek B

Kalibrační obrazec



TESTOVACÍ OBRAZEC

Dodatek C

Parametry kamery DCS-900

- Rozhraní: 10/100Mbps
- Senzor: 1/3” CMOS Digital VGA
- Barevná hloubka: 24-bit RGB
- Rozlišení: 640x480, 320x240
- Komprese: JPEG
- Počet snímků za sekundu: až 20 fps
- Minimální osvětlení: 2.5lux@f1.4, 3000K
- Uzávěrka: 1/60 ~ 1/15000 s
- Ostřící vzdálenost: 20cm ~ nekonečno
- Clona: F 1.8
- Ohnisková vzdálenost: 6.0 mm
- Napájecí napětí: DC 5V/2.5A
- Příkon: 4.5Watt (900mA x 5V)
- Provozní teplota: 5° ~ 40°C
- Provozní vlhkost vzduchu: 5% ~ 95%
- Rozměry: 2.5” (L) x 2.5" (W) x 2.75" (H)
- Váha : 0.61 lbs

Dodatek D

Ukázky korekcí zakřivení čočky

Zleva: originální obraz, transformovaný obraz s zakřivením 0.08, transformovaný obraz s zakřivením -0.08.

Dodatek E

Vzorový konfigurační soubor

```
source=video.avi
window_position=245 95
window_size=128 280
detection_line_pos=250
gui=true
dx=-2
dy=0
```

Dodatek F

Způsob umístění kamery nad linkou