Mendelova univerzita v Brně
Provozně ekonomická fakulta

Detekce polohy objektů pomocí zpracování obrazu

Diplomová práce

Vedoucí práce:
Ing. Jan Kolomazník, Ph.D.

Bc. Ladislav Koláček

Brno 2016

Na tomto místě bych rád poděkoval svému vedoucímu Ing. Janu Kolomazníkovi, Ph.D. za věcné připomínky a vstřícnost při zpracování diplomové práce. Dále bych rád poděkoval prof. RNDr. Ing. Jiřímu Šťastnému, Csc. a Ing. Martinu Šimonovi za cenné rady. Závěrečné poděkování patří mé rodině a přítelkyni za podporu, kterou mi poskytovali po celou dobu studia.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Detekce polohy objektů pomocí zpracování obrazu**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 23. května 2016 .....

Abstract

Koláček, L. *Detection of the object position using image processing.* Diploma thesis. Brno, 2016.

This thesis focuses on development of two methods, which are providing computer vision for robot. The first method is based on Hough transform and the second one is based on machine learning. The chapters in this thesis describe basic methods of image processing and object detection in image, which are closely associated with the development of the earlier mentioned methods. The thesis also includes comparison of processing time and detection rate of the implemented methods. Tests have proven that better results were obtained with the method based on Hough transform.

Keywords: Computer vision, Canny, Hough transform, machine learning, Haar, LBP, thresholding, AdaBoost.

Abstrakt

Koláček, L. *Detekce polohy objektů pomocí zpracování obrazu.* Diplomová práce. Brno, 2016.

Tato práce se zaměřuje na vývoj dvou metod zprostředkujících počítačové vidění pro robota. První metoda je založena na Houghově transformaci a druhá na strojovém učení. Jednotlivé kapitoly práce popisují základní metody zpracování obrazu a detekce objektů v obraze, které jsou úzce spjaty s vývojem výše zmíněných metod. Součástí práce je i srovnání doby zpracování a úspěšnosti detekce implementovaných metod. Testy prokázaly, že lepších výsledků bylo dosaženo při použití metody založené na Houghově transformaci.

Klíčová slova: Počítačové vidění, Canny, Houghova transformace, strojové učení, Haar, LBP, prahování, AdaBoost.

Obsah

1 Úvod a cíl práce

Tato práce se zabývá problematikou počítačového vidění. Jedná se o velmi rozsáhlé a komplikované téma. Zjednodušeně lze říci, že se počítačové vidění snaží co nejvíce přiblížit našemu lidskému zraku, neboť se pokouší z pořízeného snímku, resp. obrazu, extrahovat co největší množství v něm obsažených informací.

Počítačové vidění zahrnuje mnoho různých principů, jako příklad lze uvést detekce a sledování objektů v obraze, konstrukce modelů scény na základě několika snímků, určení polohy kamery a mnoho dalších.

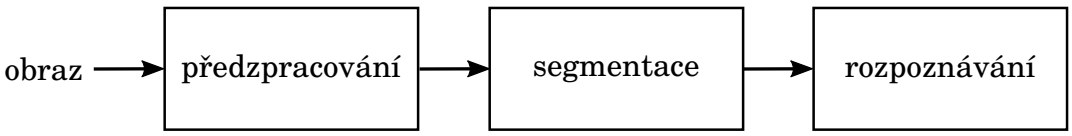
Počítačové vidění je důležitým oborem především díky jeho aplikacím v praxi. V průmyslu lze např. využít ke kontrole kvality výrobků, tedy jestli tvar a povrch odpovídá specifikaci, dále ho lze využít jako bezpečnostní prvek v automobilech, kde je vozidlo schopné autonomně zastavit, pokud hrozí kolize s detekovanou překážkou atd.

Cílem mé práce je zkonstruovat počítačové vidění pro robota, jež se má zúčastnit soutěže ve sběru plechovek v hracím poli. Více informací o této soutěži a o samotném robotovi je zmíněno v sekci Ketchup House. V této práci se převážně zaměřím na detekci polohy objektů v obraze na základě jednoho kamerového snímku. Výsledkem práce bude srovnání dvou naimplementovaných metod řešících výše zmíněný problém a zhodnocení, která metoda je více vhodná pro nasazení do provozu. První metoda bude založena na Houghových transformacích přímek a druhá na strojovém učení. Obě tyto metody budou popsány v kapitole Metodika práce.

V dalších kapitolách se budu věnovat popisu metod zpracování obrazu, které úzce souvisí s řešením daného problému. Dále zde budou popsány implementační detaily obou metod, jejich srovnání na základě několika metrik a závěrečné zhodnocení s návrhem možných rozšíření práce.

2 Zpracování obrazu a detekce objektů v obraze

Tato kapitola se zabývá zpracováním obrazu a detailně popisuje některé metody, které jsou využívány při detekci objektů v obraze. Zpracování obrazu se skládá z několika kroků, které jsou zobrazeny v diagramu na Obrázku 1.



Obrázek 1: Diagram popisuje všeobecné kroky při zpracování obrazu.

Obvykle se nejprve provádí předzpracování obrazu. Jedná se o zcela zásadní operaci, jelikož pořízený obraz často obsahuje nežádoucí jevy v podobě jakéhokoliv rušení či šumu, ke kterým dochází například při pořizování snímku ve špatně osvětlených podmínkách či kompresi obrazu. Všechny tyto problémy lze do jisté míry potlačit. Operace, které se na to používají, je nepřeberné množství, jmenovitě lze zmínit operace k úpravě kontrastu a jasu, filtrace pro zaostření nebo grafické transformace pro posun, zmenšení, či natočení obrazu a mnoho dalších. Součástí předzpracování bývá i převod obrazu z barevného modelu (např. RGB) do jiného barevného modelu v závislosti na tom, které operace budou na obraz aplikovány. Jako vhodný barevný model se často nabízí tradiční grayscale nebo binární obraz (Hozman, 2003).

Dalším krokem zpracování bývá segmentace, nebo-li izolování objektů, resp. částí objektů, od zbytku dat v obraze. Důvod, proč se o to pokoušíme, lze popsat na příkladě z bezpečnostního kamerového záznamu. Dlouhé hodiny bude kamera zabírat neměnný prostor, který pro nás nepřináší žádnou informaci. Zajímavé jsou pro nás takové momenty, kdy se v obraze vyskytne osoba, auto či jiný objekt, který chceme detekovat a pořídit o něm záznam z kamery. Segmentace obrazu se snaží izolovat právě tyto objekty od zbytku scény a v případě našeho příkladu ušetřit nekonečné hodiny kamerového záznamu (Bradski a Kaehler, 2008).

Kromě detekce objektů v neměnném prostředí se segmentace obrazu dále využívá k omezení prostoru, ve kterém vyhledáváme objekty z důvodu vysoké náročnosti na výpočetní výkon při zpracování obrazu. Z tohoto důvodu je žádoucí, aby zpracovávaná oblast v obraze byla co možná nejmenší. Nadbytečné informace, jakými mohou být např. snímané objekty za hrací plochou, jsou automaticky zahazovány. Dále je nutné zaměřit se na oblasti zájmu (region of interest). Pokud chceme např. v obraze vyhledat postavu a jsme schopni detekovat barvu pleti, obrys těla či kteroukoli část jejího těla, je dobré na základě této informace omezit prostor, který bude následně zpracováván (Bradski a Kaehler, 2008).

Nejtypičtější metodou segmentace je prahování, které je detailněji popsáno v sekci 2.3. Další metody, které lze využít k segmentaci obrazu, jsou detekce přímek, kružnic, dále metody pro zvýraznění hran v obraze a mnoho dalších. Některé z nich budou popsány v následujících sekcích.

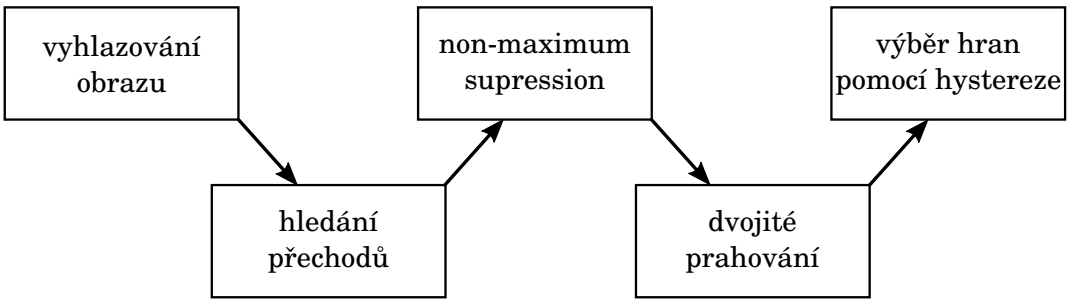
Finálním krokem zpracování obrazu bývá samotné rozpoznání objektu, které se řadí mezi náročnější části celého procesu. Zde je nutné se často zaměřit na detaily hledaného objektu, kterými mohou být např. odlišné tvarové charakteristiky. Výstupem obvykle bývá sada hodnot (mohou být uspořádány do tabulky či záznamu), které jasně definují hledaný objekt (Hozman, 2003).

2.1 Cannyho hranový detektor

Cannyho hranový detektor (Canny edge detector), jak název napovídá, slouží k detekci hran v obraze. Hlavním účelem této metody je značná redukce množství dat v obraze při zachování jejich strukturálních vlastností, které mohou být využity při dalším zpracování. Existuje více implementací algoritmů, v této práci se ale zaměřím na starší metodu, která se stala standardem pro vyhledávání hran a která byla vyvinuta J. F. Cannym. Tato metoda byla navržena na základě následujících tří kritérií (Moeslund, 2009):

- Z pohledu detekce je žádoucí, aby metoda maximalizovala pravděpodobnost vyhledání reálné hrany a naopak minimalizovala vyhledání hrany falešné.
- Z pohledu lokalizace by měly být detekované hrany co nejblíže reálným hranám.
- Z pohledu počtu vyhledaných hran by se mělo jednat o jednu detekovanou hranu, která přísluší jedné reálné hraně. Případná duplicita je nežádoucí.

Algoritmus detekce hran se skládá z pěti nezávislých kroků, jejichž výčet je zobrazen v diagramu na Obrázku 2 a které budou blíže specifikovány v následujících podsekcích (Moeslund, 2009).



Obrázek 2: Bližší pohled na proces zpracování obrazu Cannyho hranovým detektorem.

Vyhlazování

Nejprve se provádí vyhlazování obrazu za účelem potlačení šumu, např. pomocí Gaussova rozmazání (Gaussian Blurring). Ukázka původního a vyhlazeného obrazu je demonstrována na Obrázcích 3a a 3b (Canny, 2008) (Moeslund, 2009) (Bradski, 2000).

(a) Původní obrázek (b) Vyhlazený obrázek

Obrázek 3: K demonstraci Gaussova rozmazání byl využit obrázek Lenna. Na Obrázku 3a se nachází neupravený obrázek a na Obrázku 3b je demonstrován výše zmíněný efekt rozmazání, který pomáhá odstranit nežádoucí šum v obraze.

Hledání přechodů

Ve druhém kroku se provádí hledání přechodů (gradientů) v místech, kde se nachází silný okraj hrany (místa obrazu s vyššími magnitudami). K hledání přechodů se používají dvě konvoluční masky o rozměrech 3×3. Ukázka obrazu se zvýrazněnými přechody aplikováním Sobel operátoru je demonstrována na Obrázku 4 (Green, 2002).

Non-maximum supression

Ve třetím kroku se provádí „potlačení“ (non-maximum supression). Účelem tohoto kroku je provést „zaostření rozmazaných hran“, které vznikly v počátečním kroku vyhlazování v místech, kde se nachází vysoké magnitudy. Nastínění problému je takové, že se ponechají v obraze lokální maxima a všechno ostatní bude odstraněno (Canny, 2008) (Moeslund, 2009).

Dvojité prahování

Ve čtvrtém kroku se provádí dvojité prahování. Po třetím kroku je v obraze vyznačeno mnoho hranových pixelů. Může se stát, že některé z nich nenáleží skutečné hraně, ale byly vyhledány zapříčiněním šumu v obraze, či mohou být způsobeny hrubým povrchem anebo jinými nežádoucími jevy v obraze. Z tohoto důvodu se provádí dvojité prahování. První „hranice“ prahování se nastaví na velmi vysokou hodnotu (např. 80 %) a má za úkol zvýraznit velmi silné hrany v obraze. Druhé prahování

Obrázek 4: Zvýrazněné přechody (gradienty) získané aplikováním sobel operátoru.

je nastaveno na podstatně nižší hodnotu (např. 20 %). Pixely, které nespadají do této hranice, jsou automaticky potlačeny a ostatní pixely budou označeny jako slabé hrany (Canny, 2008) (Moeslund, 2009).

Výběr hran pomocí hystereze

Ve finálním kroku se vyberou hrany pomocí hystereze (edge tracking by hysteresis). Ve čtvrtém kroku jsme získali silné a slabé hrany. O silných hranách jsme přesvědčeni, že skutečně jsou hranami. Ze slabých hran vybereme pouze ty hrany, které jsou spojeny se silnou hranou. U těchto hran je menší pravděpodobnost, že jsou šumem či jiným nežádoucím jevem v obraze. Výsledný obraz po aplikování všech zmíněných kroků je zobrazen v Obrázku 5 (Canny, 2008) (Moeslund, 2009).

2.2 Gaussian Blur

Jedním z mnoha algoritmů, které implementují „rozmazání“ obrazu, je Gaussian Blurring. Ke zpracování obrazu se přitom využívá Gaussovo rozdělení (Gaussian distribution).

Pojem rozmazání lze zjednodušeně chápat jako výběr hodnoty pixelu na základě průměrné hodnoty jeho okolních pixelů. Z pohledu grafiky se tento jev dá nazvat „rozmazání“ (blurring effect). Z hlediska hodnot se jedná o vyhlazování (smoothing). Nezávisle na označení, středový bod vždy přichází o určitou informaci (ztrácí se jeho detail).

Při tomto postupu není vhodné použít průměrnou hodnotu z okolních pixelů, jelikož obraz je spojitý a pixely blízko sebe jsou velmi „podobné“. Lepším řešením

Obrázek 5: Demonstrace obrazu po aplikaci posledního kroku Cannyho detekce hran. Konkrétně v tomto případě byl v průběhu procesu využit ve čtvrtém kroku dvojité prahování s hodnotami 30 a 85.

je použít vážený průměr, jelikož čím blíže jsou pixely umístěny, tím větší bude váha (PixelsTech, 2012).

Akceptovatelným modelem rozložení vah je normální (Gaussovo) rozdělení. V grafice se jedná o křivku ve tvaru zvonu, která popisuje přesně takové rozložení hodnot, jaké je popsáno výše.

Graf popisuje pouze jednodimenzionální normální rozdělení. V obraze je nutné použít dvoudimenzionální normální rozdělení.

Hustota funkce normálního rozdělení se nazývá Gaussova funkce. Jednodimenzionální tvar Gaussovy funkce je zobrazen v rovnici 1. Z této rovnice lze odvodit dvoudimenzionální tvar Gaussovy funkce, který je popsán rovnicí 2. Pomocí této rovnice lze spočítat váhu každého bodu v obraze (Fisher, Perkins, Walker a Wolfart, 2003).

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}, \tag{1}$$

$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

Symbol σ představuje v rovnicích 1 a 2 rozptyl.

2.3 Prahování

Prahování (thresholding) je nelineární operací, která se nejčastěji aplikuje na obrázek ve stupních šedi (grayscale image) a převede jej na binární obraz. Binární obraz se

bude skládat z pixelů, jejichž hodnota bude stanovena na základě prahové hodnoty (threshold value) (WaveMetrics, 2015).

V základní implementaci prahování se porovnává původní hodnota pixelu s prahovou hodnotou a pokud je tato hodnota vyšší, než je prahová hodnota, přiřadíme pixelu bílou barvu (hodnota 255), v ostatních případech přiřadíme pixelu černou barvu (hodnota 0) (WaveMetrics, 2015) (Wikipedia, 2016).

Základním předpokladem pro správnou funkci prahování je určit vhodnou prahovou hodnotu. Hlavním problémem však je, že stanovení této hodnoty probíhá na základě intenzity, nikoliv na vztahu okolních pixelů. Může tedy dojít k situaci, kdy některé izolované pixely nebudou zahrnuty do požadované oblasti. Zejména k tomuto jevu dochází v blízkosti hranic hledané oblasti. Ještě horší situace nastává, pokud obraz obsahuje šum. Intenzita obrazového bodu v zašuměné oblasti pak nemusí představovat běžnou intenzitu bodu v hledané oblasti.

Pokud tedy využijeme základní implementaci prahování, je nutné počítat s tím, že při nevhodném nastavení prahové hodnoty nebo přítomnosti mnoha různých intenzit světla v obraze, může vyprahováním docházet k odstranění důležité informace z obrazu, anebo naopak se ve výsledném obrazu vyskytne příliš mnoho nadbytečné informace (Turkel, 2011). Tento problém se do jisté míry snaží odstranit adaptivní prahování.

Adaptivní prahování

V minulé sekci bylo zmíněno, že pokud se v obraze vyskytuje mnoho variací intenzity pozadí, může dosáhnout adaptivní prahování lepších výsledků.

Na rozdíl od (globálního) prahování počítá adaptivní prahování prahovou hodnotu pro každý pixel zvlášť na základě okolních pixelů (neighborhoods). Prahovou hodnotou je vážený průměr z okolních pixelů, ze kterého se odečte offset (scikit-image, 2016).

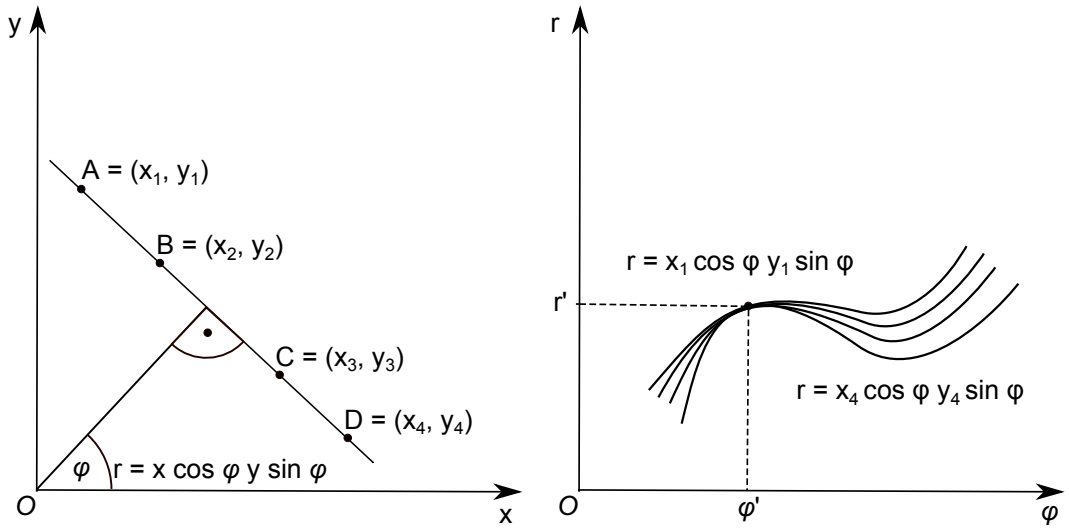
Knihovna OpenCV obsahuje několik různých implementací výpočtu prahovací hodnoty, například tyto:

- `ADAPTIVE_THRESH_MEAN_C` - výpočet na základě střední hodnoty okolí pixelu.
- `ADAPTIVE_THRESH_GAUSSIAN_C` - prahovací hodnotou je vážený součet hodnot okolních pixelů (Bradski, 2000).

Při testování bylo zjištěno, že použití `ADAPTIVE_THRESH_GAUSSIAN_C` vede na lepší výsledky s minimálním výskytem šumu v obraze, ale doba zpracování je zhruba šestkrát pomalejší než při použití `ADAPTIVE_THRESH_MEAN_C`. Z tohoto důvodu byl pro výslednou metodu vybrán výpočet prahovací hodnoty právě na základě `ADAPTIVE_THRESH_MEAN_C`, která podává velmi dobré výsledky ve velmi krátkém čase.

2.4 Houghovy transformace

Houghova transformace je metoda pro hledání přímek, kružnic, elips a jiných jednoduchých grafických útvarů v obraze. Jejím hlavním cílem je tedy segmentace objektů v obraze (Šafařík, 2011). Houghova transformace je založená na vyhledání takové množiny bodů v obraze, které leží na jedné přímce, resp. na části přímky (Perďoch, 2008). Na Obrázku 6a jsou zobrazeny body A, B, C a D. Každý z těchto bodů je



(a) Původní prostor (b) Parametrický prostor

Obrázek 6: Přímka zobrazená v původním i parametrickém prostoru.

popsán souřadnicemi (x_i, y_i), kde $i = 1..4$. Hledanou přímku lze vyjádřit ve dvourozměrném prostoru několika způsoby. Lze např. využít směrnicový tvar $y = kx + q$, kde k je směrnicí přímky a q je posun ve směru osy y. Tento tvar se v Houghově transformaci nevyužívá, jelikož pokud bychom měli definovat přímku, která je rovnoběžná s osou y, pak by se hodnota q blížila k nekonečnu. Z tohoto důvodu se používá normálový tvar přímky, který vyjádříme pomocí polárních souřadnic následujícím způsobem (Vlach, 2011) (Duda a Hart, 1972) (Canny, 2008) (Šafařík, 2011):

$$r = x \ \cos \varphi + y \ \sin \varphi, \tag{3}$$

kde r je délka normály a φ je velikost orientovaného úhlu mezi osou x a normálou. Výpočet r a φ vypadá následovně:

$$r = \sqrt{x^2 + y^2}, \tag{4}$$

$$\varphi = \arctan \frac{y}{x} \tag{5}$$

Výhodou této metody jsou její dobré výsledky detekce i přes případné nepravidelnosti a poruchy v hledaných křivkách. Tyto nedokonalosti mohou vzniknout případným

šumem či při ztrátě informace v obrazových datech (např. po aplikaci prahování) (Šafařík, 2011).

Detekce přímek

Ze sekce Houghovy transformace vyplývá, že k nalezení přímek bude nutné použít parametrický prostor. Příklad převodu do tohoto prostoru lze demonstrovat na bodu A z Obrázku 6a. Vstupními parametry pro nás budou souřadnice tohoto bodu. Po jejich dosazení do rovnice 3 získáme normálový tvar přímky (6), která prochází bodem A.

$$r = x_1 \cos \varphi + y_1 \sin \varphi. \tag{6}$$

Parametry r a φ jsou pro nás v této rovnici neznámými. Pokud proložíme tímto bodem všechny proveditelné přímky dosazením a následným dopočítáním dvojice (φ, r), tak získáme v parametrickém prostoru křivku, která má sinusový průběh. Ilustrační příklad takové křivky pro bod A je demonstrován v Obrázku 6b. Pokud stejný proces uplatníme i na ostatní body B, C a D, zjistíme, že se vzniklé křivky v parametrickém prostoru protnou v jediném bodě (φ, r). Tato skutečnost je rovněž ilustrována v Obrázku 6b (Canny, 2008) (Šafařík, 2011) (Perďoch, 2008).

Algoritmus Houghovy transformace přímek lze popsat následujícím způsobem: Vstupním prvkem je binární obraz f. Všechny pixely, jejichž hodnota jasu se rovná 1, převedeme do parametrického prostoru, který nazveme akumulátor A o rozměrech $M \times N$ (viz. Obrázek 6b). Pro každý pixel, jehož hodnota jasu se rovná 1, vypočítáme hodnotu parametru r dosazením parametru φ z předem dohodnutého intervalu do rovnice 3. Následně inkrementujeme hodnotu v akumulátoru na všech pozicích $A(\varphi_i, r_j) = A(\varphi_i, r_j) + 1$, pro $\forall i = 1..M$ a $\forall j = 1..N$. Po zpracování všech pixelů v obraze je akumulátor $A(\varphi_i, r_j)$ naplněn hodnotami, které značí počet nalezených bodů na jedné přímce, která je definována parametry (φ_i, r_j). Dle maximálních hodnot v akumulátoru vybereme požadovaný počet přímek v obraze, které jsou výstupem Houghovy transformace(Vlach, 2011)(Duda a Hart, 1972) (Canny, 2008).

Detekce kružnic

Pro vyhledávání ostatních geometrických útvarů, které lze analyticky popsat, je možné použít stejnou proceduru jako tu, která byla popsána v předchozí podsekci Detekce přímek. Při vyhledávání kružnice v obraze budeme vycházet z rovnice 7.

$$(x - a)^2 + (y - b)^2 = r^2, \tag{7}$$

kde a a b jsou souřadnice středu kružnice a r je poloměrem kružnice. Každý bod, který náleží takové kružnici, lze popsat pomocí rovnic 8 a 9.

$$x = a + r \cos \varphi, \tag{8}$$

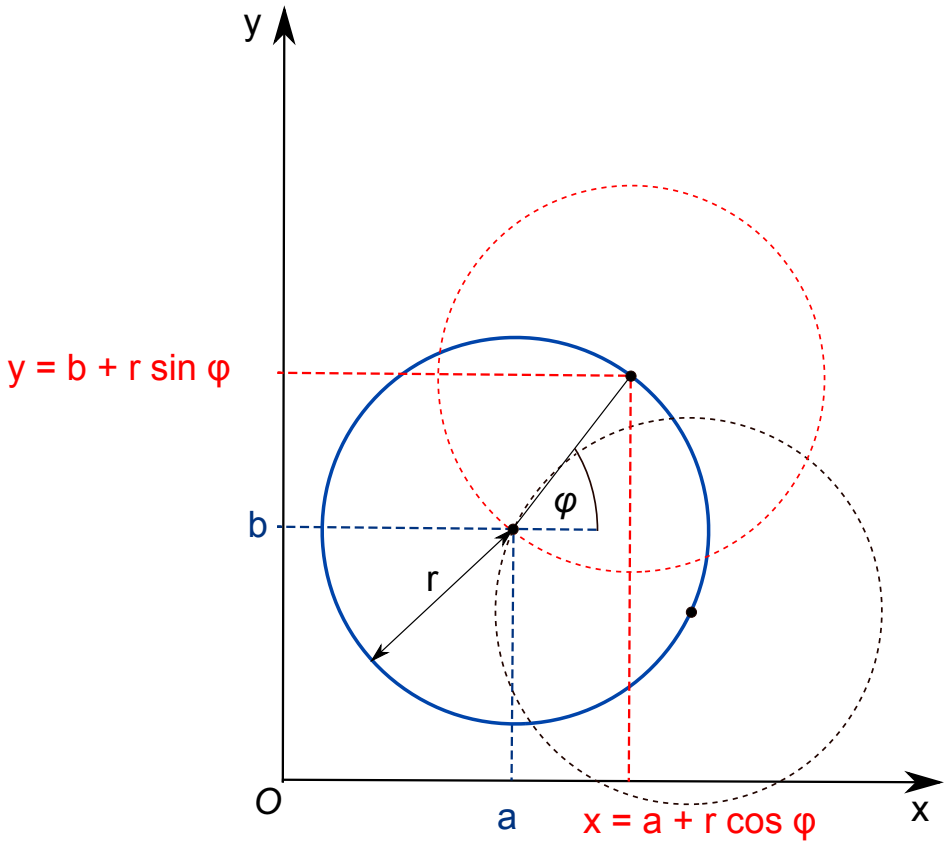
$$y = b + r \sin \varphi \tag{9}$$

Algoritmus bude vyhledávat body se společným středem kružnice daným souřadnicemi a a b, jak již bylo zmíněno výše. Výpočet souřadnic je odvozen z rovnic 8 a 9.

$$a = x - r\cos\varphi, \tag{10}$$
$$b = y - r\sin\varphi \tag{11}$$

Algoritmus Houghovy transformace kružnic lze popsat následujícím způsobem: Počátek fáze je shodný s algoritmem vyhledávání přímek. Vstupem je binární obraz f. Vybereme všechny pixely, jejichž hodnota jasu se rovná 1. Dále zvolíme poloměr r hledané kružnice. Podobně jako v předchozí sekci stanovíme akumulátor A o rozměrech $M \times N$.

Pro každý pixel, jehož hodnota jasu se rovná 1, vypočítáme hodnoty a a b na základě rovnic 10 a 11, kde úhel $\varphi = 0$. Pokud se hodnota a bude vyskytovat v intervalu $(0, M)$ a zároveň b bude náležet do intervalu $(0, N)$, inkrementujeme hodnotu v akumulátoru $A(a, b) = A(a, b) + 1$. Po zpracování všech pixelů v obraze je akumulátor naplněn hodnotami, které značí počet kružnic se středem v (x_i, y_j) a poloměrem r. Dle maximální hodnoty v akumulátoru vybereme střed (x_i, y_j) hledané kružnice. Princip této metody je naznačen na Obrázku 7. (Vlach, 2011) (Duda a Hart, 1972) (Canny, 2008)



Obrázek 7: Na obrázku jsou zobrazeny tři kružnice. Modrá kružnice je hledanou kružnicí s poloměrem r, kterou vyhledáváme pomocí Houghovy transformace. Zbylé dvě kružnice značí okolí bodů ve vzdálenosti r, ve kterém hledáme střed požadované kružnice.

3 Metody strojového učení

Cílem metod strojového učení je převedení dat na informace. Hlavně se však snaží o jejich porozumění. K tomu je nutné provést učení na velké množině trénovacích dat. Tímto způsobem získáme natrénovaný stroj, resp. metodu, která by měla být schopna poskytnout o datech bližší informace. Takovou informací může být například tvrzení o tom, jestli se hledaný objekt vyskytuje v obraze. Na základě této informace lze např. provést vyhledání polohy objektu a jeho vyznačení v obraze či kamerovém snímku. Dále je možné omezit prostor pro vyhledávání dalších objektů jako v případě obličeje, kde se snažíme upřesnit polohu očí, nosu nebo úst. V případě strojového učení se výše zmíněný převod dat na informace provádí extrahováním vzorů (patterns) z dat.

Strojové učení lze uplatnit na mnoho různých dat, jako jsou tržní ceny, teplota, barevná intenzita a mnoho dalších dat. Data bývají často předzpracována do tzv. příznaků (features) (Bradski a Kaehler, 2008).

V následujících sekcích bude probrána metoda Viola-Jones, která se nejčastěji využívá k detekci obličejů v obraze a která byla první metodou umožňující detekci v reálném čase. Dále budou zmíněny základní principy a algoritmy, které způsobují, že je metoda Viola-Jones tak efektivním a rychlým algoritmem. Závěrečná sekce bude věnována LBP příznakům.

3.1 Metoda Viola-Jones

Viola a Jones (2001) ve svém článku *Rapid Object Detection using a Boosted Cascade of Simple Features* představili metodu pro detekci obličejů v obraze. Tato metoda byla v té době výrazně rychlejší a přesnější než ostatní metody řešící tento problém. Konkurenční metody trpěly vysokou výpočetní náročností, jelikož vyhledávání obličejů bylo založeno na barevných tónech pleti, hledání obrysů anebo u více komplexních metod dokonce vyžadovaly zahrnutí šablon či neuronových sítí.

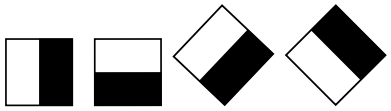
Obraz jako takový je kolekcí hodnot barev (intenzit světla). Analýza těchto pixelů je časově velmi náročná, jelikož lidský obličej může být v obraze různě natočen, dále může nabývat odlišných tvarů a nelze opomenout ani odlišnou pigmentaci kůže. Analýza pixelů často musela být prováděna opakovaně kvůli změně měřítka (scaling) a zpřesňování detekce (Wilson a Fernandez, 2006).

Metoda Violy a Jonese již není založena na práci s pixely, ale k detekci obličejů využívá Haarovy příznaky a algoritmus pro trénování klasifikátorů kaskádu AdaBoost (AdaBoost cascade). Algoritmus AdaBoost a výše zmíněné Haarovy příznaky budou detailněji popsány v následujících podsekcích (Wilson a Fernandez, 2006).

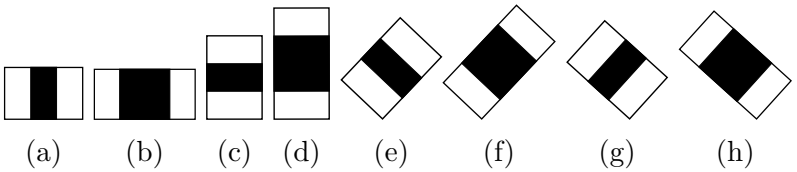
Haarovy klasifikátory

Jádrem Haarova klasifikátoru jsou Haarovy příznaky (Haar-like features).Tyto příznaky nepracují s intenzitou hodnot jednotlivých pixelů, jak je popsáno výše, ale

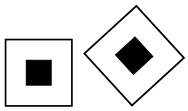
namísto toho pracují s rozdíly hodnot mezi celými skupinami pixelů (tyto skupiny pixelů tvoří obdelníkovou oblast). Tímto způsobem lze snadno v obraze rozlišit světlé oblasti od tmavých. Další výhodou Haarových příznaků je jejich snadná škálovatelnost, která se provádí prostým zvětšením, resp. snížením velikosti skupiny pixelů. Tímto způsobem je možné v obraze vyhledávat objekty různých velikostí. Na následujících třech obrázcích jsou zobrazeny různé typy Haarových příznaků (Wilson a Fernandez, 2006):



Obrázek 8: Hranové Haarovy příznaky.



Obrázek 9: Čárové (line) Haarovy příznaky.



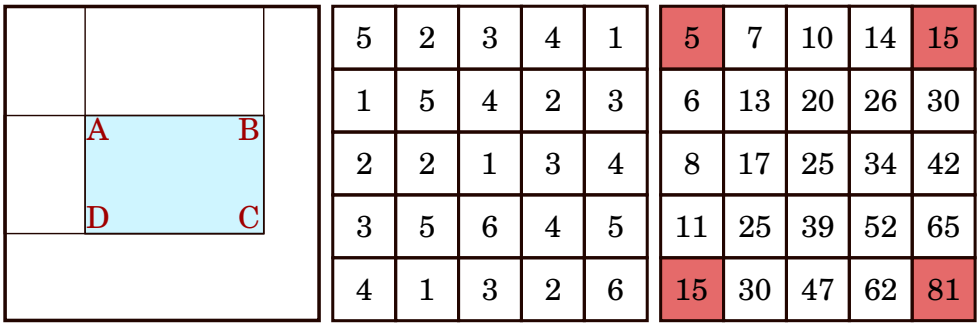
Obrázek 10: Středové (center-surround) Haarovy příznaky.

Integrální obraz

Integrální obraz je pole hodnot sestávající ze součtů intenzit pixelů. Hodnotu v poli na pozici (x, y) získáme součtem intenzit pixelů vlevo a nahoře od této pozice a následným přičtením hodnoty intenzity pixelu na této pozici. Pokud budeme uvažovat, že $A[x, y]$ je pozice pixelu v původním obraze a $AI[x', y']$ je pozicí v integrálním obraze, tak výpočet integrálního obrazu lze popsat rovnicí 12 (Wilson a Fernandez, 2006).

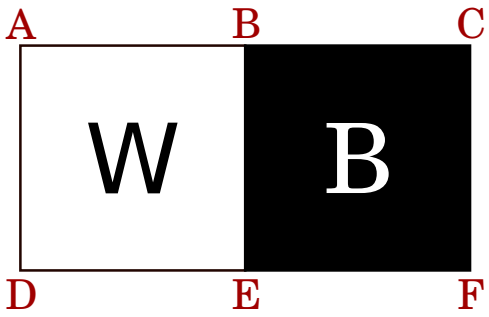
$$AI[x', y'] \quad = \sum_{x' \leq x,\ y' \leq y} A[x, y] \tag{12}$$

Ukázka integrálního obrazu je demonstrována v Obrázku 11 společně s příkladem, jak spočítat sumu pixelů libovolné části obrazu pouze se čtyřmi přístupy do paměti.



(a) (b) (c)

Obrázek 11: Na Obrázku 11a je zobrazeno, jak probíhá výběr podobrázku při zpracování celého obrazu, a na Obrázcích 11b a 11c je demonstrován výpočet sumy pixelů plochy podobrázku pomocí 4 krajních bodů, které jsou vyznačeny v Obrázcích 11a a 11c. Výpočet pak vypadá následovně: $S = A + C - B - D$

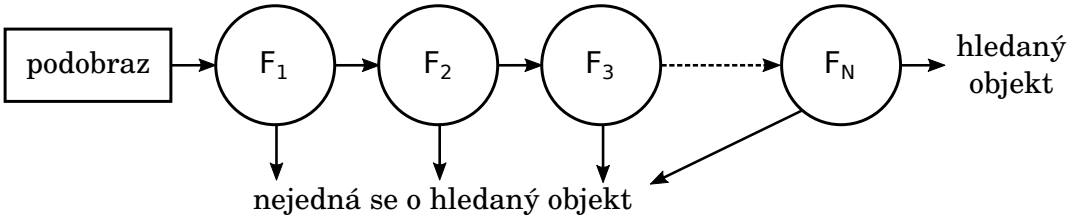


Obrázek 12: Na obrázku je vysvětleno, proč je výpočet Haarových příznaků velmi rychlý a efektivní. Jedná se o výpočet rozdílu dvou integrálních obrazů (černé a bílé plochy), který se skládá z jednoduchých aritmetických operací a několika přístupů do paměti. Výslednou rovnici pro výpočet hranového příznaku, který je vidět na obrázku, lze vyjádřit následovně: $S = A - 2B + 2E - D - F + C$

Kaskáda klasifikátorů

Výpočet příznaku je navržen efektivně a je velmi rychlý, ale výpočet všech příznaků ve zpracovávaném podobrázku (jedná se o část zpracovávaného obrazu o velikosti např. 24×24 pixelů) je nesmyslný, protože tato hodnota se může pohybovat i ve stovkách tisíc. K tomu, aby bylo možné rozhodnout, jestli podobrázek zahrnuje požadovaný objekt, postačí spočítat jen malou část výše zmíněných příznaků. Z tohoto vyplývá, že úkolem kaskády je efektivně eliminovat 50 % a více podobrázků, ve kterých se hledaný objekt s jistotou nevyskytuje s využitím jen minimální režie na výpočetní zdroje. Takové podobrázky jsou eliminovány již při analýze několika prvních klasifikátorů v kaskádě, jak je zobrazeno v Obrázku 13. Pokud při analýze podobrázek nevyhovuje klasifikátoru, okamžitě je zahozen a zpracovává se následující podobrázek. Tímto způsobem je algoritmus schopen velmi rychle zpracovat celý

obrázek a navíc s velkou přesností detekovat hledaný objekt (Wilson a Fernandez, 2006).



Obrázek 13: Analýzy podobrazu pomocí kaskády klasifikátorů. V Obraze je demonstrováno zahazování podobrazu, ve kterém se s jistotou daný objekt nevyskytuje.

AdaBoost

Adaptivní boosting (AdaBoost) je algoritmus, který se používá k natrénování několika slabých klasifikátorů h_t. Jednotlivě bývají tyto klasifikátory velmi jednoduché. Zpravidla se totiž jedná o rozhodovací stromy s jediným dělením (např. na základě jedné prahové hodnoty), které se nazývají desicion stumps. Každému klasifikátoru přiřazujeme váhu α na základě chyby, které se dopustil při klasifikaci. Tato chyba nesmí být vyšší než hodnota 0.5, protože takový klasifikátor by neposkytoval žádnou užitečnou informaci (Bradski a Kaehler, 2008).

Zpracovaná data v podobě příznakových vektorů x_i je nutné ohodnotit. V AdaBoostu se využívá binární klasifikace dat, kde třída y_i nabývá hodnot $-1, +1$, které značí přítomnost, anebo naopak nepřítomnost hledaného objektu (Bradski a Kaehler, 2008).

Cílem boostingu je zvolit několik slabých klasifikátorů a následně je „spojit“ do jediného klasifikátoru, který bude nazýván silným klasifikátorem. Tedy je nutné vytvořit lineární kombinaci slabých klasifikátorů.

Nyní je možné zadefinovat pojem distribuční funkce D_t nad daty, kde $D_t(i)$ vyjadřuje váhu pro každý příznakový vektor x_i. Součet těchto vah dává dohromady hodnotu 1. AdaBoost je iteračním algoritmem. V každé iteraci se na trénovacích datech natrénuje jeden slabý klasifikátor a následně je přidán do silného klasifikátoru. Na základě těchto vah je algoritmus schopen se v další iteraci zaměřit na vzorky dat, ve kterých se klasifikátor dopouštěl nejvíce chyb a snaží se vytvořit klasifikátor, který tyto vzorky dat bude klasifikovat správně. Po mnoha iteracích je důsledkem, že všechna trénovací data jsou klasifikovány bezchybně. Trénovací chyba klesá exponenciálně rychle. Pokud chyba na trénovacích datech klesne na nulu, stále má smysl provádět další trénovaní, jelikož je možné dále snižovat chybu v testovacích datech (Bradski a Kaehler, 2008).

Oproti jiným algoritmům, jako jsou např. neuronové sítě, AdaBoost není náchylný na přetrénování. Pokud se v trénovacích datech nevyskytuje šum, chyba na

testovacích datech stále klesá, popř. stagnuje.

Algorithm 1: Pseudokód algoritmu AdaBoost (Bradski a Kaehler, 2008).

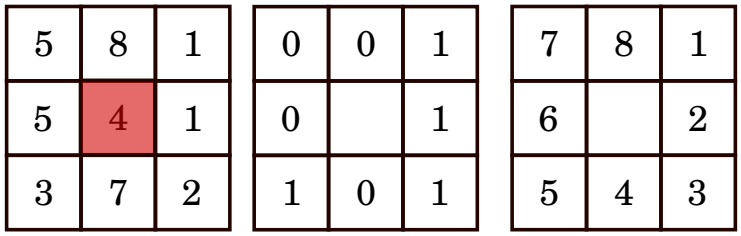
Data: Vzorky dat
Result: Natrénování silného klasifikátoru
1 inicializace;
2 $D_1(i) = 1/m, i = 1..m$;
3 **for** *t=1..T* **do**
4 nalezni klasifikátor h_t, který minimalizuje váhovou chybu $D_t(i)$;
5 $h_t = \arg\min_{h_j \in H} \epsilon_j$, kde $\epsilon_j = \sum_{i=1}^{m} D_t(i)$ dokud $\epsilon_j < 0.5$;
6 jinak ukonči.
7 Nastav váhu $\alpha_t = \frac{1}{2}\log[(1-\epsilon_t)/\epsilon_t]$, kde ϵ_t je minimální chyba z kroku 5.
8 Aktualizuj váhy dat: $D_{t+1}(i) = [D_t(i)\exp(-\alpha_t y_i h_t(x_i))]/Z_t$, kde Z_t normalizuje výpočet přes všechna data.
9 **end**

3.2 LBP

LBP (Local binary pattern) je jednoduchým, ale velmi účinným obrazovým operátorem, který označuje každý pixel v obraze hodnotou, jež se určí ze sousedních pixelů. Díky nízké výpočetní náročnosti byl využit v mnoha aplikacích, kde bývá nejčastěji aplikován na obraz ve stupních šedi (Pietikäinen, 2010).

Základní LBP operátor, který představili autoři Ojala, Pietikäinen a Harwood (1996) v článku *A comparative study of texture measures with classification based on featured distributions*, přiřazuje hodnoty pixelům v obraze z bloku pixelů o velikosti 3×3 pixely. Pixely v tomto bloku jsou vyprahovány hodnotou středového pixelu. Pokud mají pixely nižší intenzitu než středový pixel, jsou ohodnoceny 1, v opačném případě 0. Vyprahování pixelů je demonstrováno na Obrázcích 14a a 14b. Tyto hodnoty je nutné přepsat do 8bitového čísla. Přepis hodnot se odvíjí od výběru prvního pixelu a následně se provádí dokola dle nebo proti směru hodinových ručiček, jak je zobrazeno na Obrázku 14c. Nezáleží na tom, kterým pixelem začneme sousední pixely přepisovat, ale je důležité, abychom tohle pořadí zachovali pro všechny ostatní zpracovávané pixely v obraze (Rosebrock, 2015) (Pietikäinen, 2010) (Pietikäinen, Hadid, Zhao a Ahonen, 2011).

Binární číslo je následně konvertováno do decimálního tvaru, který již představuje výše zmíněnou LBP hodnotu středového pixelu. Jelikož se sousedství středového pixelu skládá z 8 pixelů, získáváme dohromady až $2^8 = 256$ kombinací hodnot pro pixely (Pietikäinen, 2010) (Pietikäinen, Hadid, Zhao a Ahonen, 2011). Z četnosti výskytů jednotlivých LBP hodnot lze sestrojit histogram, který představuje náš výsledný příznakový vektor (Rosebrock, 2015). Histogram je rovněž možné použít ke klasifikaci textur v obraze (scikit-image, 2016).



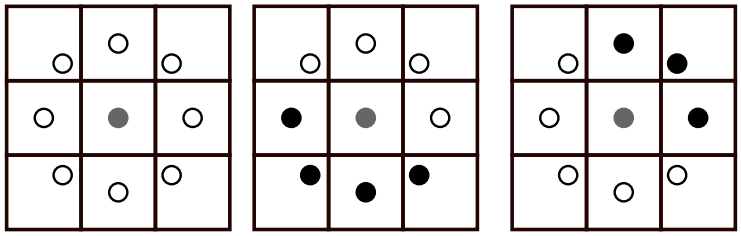
(a) Vzorky (b) Prahování (c) Přepis hodnot

Obrázek 14: Na obrázcích 14a a 14b je demonstrováno vyprahování vzorků ze sousedství červeně zvýrazněného pixelu. Přepisem hodnot v Obrázku 14b dle pořadí z Obrázku 14c získáme binární hodnotu 00010111. Označení pixelu získáme výpočtem $1 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 23$

O pár let později byl LBP rozšířen o podporu různě velkých sousedství pixelu. S tímto rozšířením byly představeny následující dva parametry (Rosebrock, 2015).

- Počet souměrně umístěných bodů p v kruhovém sousedství.
- Poloměr kružnice r.

U LBP dále uvažujeme pojem uniformita (uniformity). LBP je považováno za uniformní pokud, pokud jeho 8bitové číslo obsahovalo nejvýše dva přechody $0 - 1$ nebo $1 - 0$. Například binární číslo 00100000 je uniformní, protože se v něm vyskytují pouze dva přechody. Této znalosti se dále využívá při určování hran a rohů v lokálních částech obrazu, jak je demonstrováno v Obrázku 15 (Rosebrock, 2015).



(a) Plocha (b) Hrana (c) Roh

Obrázek 15: Na obrázcích jsou demonstrovány pixely s nižší, resp. vyšší intenzitou než středový pixel což reprezentují černé, resp. bílé tečky. Černá tečka zde odpovídá binární hodnotě 1.

4 Metodika práce

V této části budou popsány kroky, které jsem učinil k vyřešení problému detekce plechovek na hrací ploše v rámci soutěže Ketchup House. Tato soutěž bude popsána v následující sekci. Než byly vybrány dvě finální metody, u kterých bylo vyhodnoceno, že jsou schopny vyřešit daný problém v reálném čase s podáním velmi přesných výsledků, bylo vyvinuto mnoho jednorázových prototypů, které byly po určité době zahozeny jako nepřesné, pomalé či nespolehlivé. V době návrhu se při výběru metod vycházelo z dostupných prostředků robota, pro kterého je vyvíjené počítačové vidění určeno.

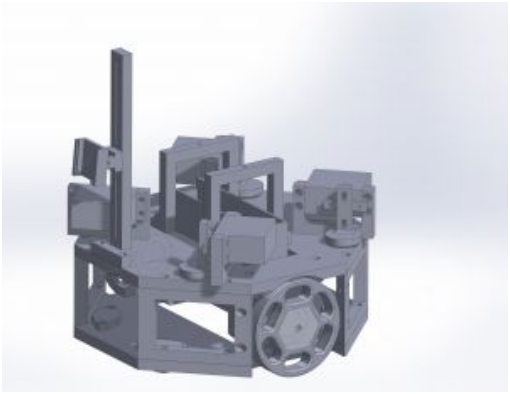
Původně měl robot disponovat nejen kamerou, ale i laserovým ukazovátkem. Tento laser měl vytvořit na snímané plechovce vzor, který by bylo možné efektivně a rychle detekovat. Laser však nebyl dostatečně výkonný, a proto bylo od tohoto řešení upuštěno. Jelikož bylo možné využít pouze kamerová data, rozhodl jsem se první metodu založit na strojovém učení. Po natrénování klasifikátoru na velké množině dat je metoda schopná podat výborné výsledky detekce v prakticky libovolném prostředí. Více informací o strojovém učení je popsáno v kapitole Metody strojového učení. Druhá metoda byla zvolena na základě navržení hrací plochy, které přímo vybízí k využití Houghovy transformace přímek a tedy k efektivní segmentaci obrazu. Více informací o Houghových transformacích je uvedeno v sekci Houghovy transformace. V následujících sekcích jsou tyto dvě metody detailněji popsány společně s pravidly pro soutěž Ketchup House.

4.1 Ketchup House

Ketchup House je soutěž, kde dvojice robotů soutěží ve sběru plechovek. Vítězným robotem se stává ten, který jich má po tří minutovém časovém intervalu ve skladu více.

Hřiště je tvořeno z 5 vodorovných a 5 svislých černě zbarvených čar, které vzájemně tvoří síť. Rozestup mezi sousedními čarami je 30 cm. Zbytek hrací plochy tvoří bílá barva. Protilehlé krajní čáry hracího pole symbolizují sklady robotů, kam odkládají plechovky. Plechovka je uznána pouze tehdy, pokud se po umístění alespoň částečně dotýká skladu. Plechovky jsou umístěny pouze na průsečících svislých a vodorovných přímek. Dvě plechovky jsou umístěny vždy na stejném místě na středové čáře mezi sklady, další tři plechovky jsou náhodně rozmístěny na ostatních průsečících. Pokud se robot s plechovkou vzdálí o více jak jeden čtverec, je na její místo dosazena další plechovka, a to až do maximálního počtu 12.

Omezení se netýká jen hřiště a rozmístění plechovek, ale i robota a samotné plechovky. Robot nesmí být okolí nebezpečný a musí fungovat zcela autonomně. Pro plechovku platí, že se musí jednat o volně dostupnou plechovku od kečupu. Oba objekty jsou pak omezeny rozměrově. Robot je omezen podstavou o velikosti maximálně 30×30 cm a plechovka průměrem o velikosti 53 mm a výškou 74 mm. Model robota je zobrazen na Obrázku 16.



Obrázek 16: Mobilní robot K4 určený do soutěže Ketchup House (Tým AiStorm, 2016).

4.2 Metoda založená na Houghově transformaci

Metoda založená na Houghově transformaci vyžaduje, abychom se více zaměřit na předzpracování obrázku. Před použitím Houghovy transformace za účelem detekce přímek v obraze je důležité nejprve získat obrázek, ve kterém je potlačen šum a kde jsou maximálně zvýrazněny hrany.

Po aplikování Houghovy transformace je nutné získané přímky rozdělit do dvou skupin. První skupina bude obsahovat vodorovné přímky hrací plochy a druhá skupina svislé přímky této plochy. Více informací o hrací ploše je uvedeno v sekci 4.1.

Dále je nutné vyhledat průsečíky výše zmíněných skupin přímek. U každého průsečíku bude stanovena oblast, ve které budeme detekovat plechovku na základě poměru černých a bílých pixelů v této oblasti.

Na závěr se u detekovaných plechovek dopočítá jejich pozice v polární souřadnicích vzhledem k robotovi na základě pozice plechovky v obraze a její výšky v pixelech.

4.3 Metoda založená na strojovém učení

V této sekci bude popsán výběr vhodné metody strojového učení a dále zde bude nastíněno, jak se takové učení provádí. Výběr vhodné metody byl založen na srovnání úspěšnosti detekce různých objektů v obraze na základě již existujících analýz. Na základě těchto analýz bylo zjištěno, že velmi dobrých výsledků detekce objektů dosahují detektory využívající natrénovanou kaskádu klasifikátorů, kde tyto klasifikátory využívají různé typy příznaků.

Kaskáda slabých klasifikátorů využívající Haarovy příznaky dosahuje velmi dobrých výsledků při detekci obličejů (zároveň i nosu, očí a úst). Velmi dobrých výsledků (úspěšnost vyšší než 71 %) bylo dosaženo i při detekci dopravních značek, jak uvádějí autoři Doman, Deguchi, Takahashi, Mekada, Ide a Murase (2009) v článku *Construction of Cascaded Traffic Sign Detector Using Generative Learning*. Další

srovnání provedli autoři Han, Han a Hahn (2009) v článku *Vehicle Detection Method using Haar-like Feature on Real Time System*, ve kterém otestovali úspěšnost detekce před námi se vyskytujících vozidel. Úspěšnost detekce zde přesáhla 75 %.

Další srovnání úspěšnosti detekce objektu (objektem byl kmen stromu), které publikovali autoři Jiang, Hornegger, Koch (2014) v knize *Pattern Recognition*, bylo zaměřeno na kaskády slabých klasifikátorů, které využívají různé typy příznaků. Lepších výsledků úspěšnosti detekce zde bylo dosaženo s využitím LBP příznaků (95.8 %) a Haar příznaků (95.1 %) oproti HOG příznakům (77.9 %).

Na základě výše zmíněných analýz byla k tréninku zvolena kaskáda slabých klasifikátorů s využitím příznaků LBP i Haar, avšak k závěrečnému srovnání bude zvolena jen ta kaskáda, která bude dosahovat lepších výsledků detekce objektů.

Natrénování kaskády klasifikátorů vyžaduje pořízení dostatečně velké sady fotek hledaného objektu. V našem případě je vyhledávaným objektem plechovka od kečupu. Z pořízených fotek je nutné sestavit trénovací sadu, na které dojde k naučení kaskády slabých klasifikátorů. Ze zbylých fotek vytvoříme testovací sadu, na které bude ověřována úspěšnost detekce natrénovaného klasifikátoru.

Dále je nutné pořídit sadu negativních snímků. Negativní snímky jsou takové snímky, na kterých se nevyskytuje hledaný objekt. Tahle sada fotek by měla být výrazně větší než je sada trénovacích dat. Všechny snímky byly pořízeny výše zmíněným mobilním robotem K4. Na závěr je nutné naimplementovat metodu, která bude využívat výše zmíněný klasifikátor.

5 Implementace metod

Pro implementace metod počítačového vidění robota bude použit programovací jazyk Python verze 2.7 s využitím metod z knihovny OpenCV verze 2.4.9 a NumPy verze 1.8.2.

OpenCV (Open Source Computer Vision) poskytuje více jak 2500 optimalizovaných algoritmů především pro zpracování počítačového vidění v reálném čase (Bradski, 2000).

Numpy poskytuje metody pro vědecké výpočty prováděné v Pythonu. Jsou zde zahrnuty operace s objekty N-dimenzionálních polí, lineární algebra, Fourierovy transformace a mnoho dalších (NumPy, 2005).

Pomocné skripty budou implementovány ve skriptovacím jazyce bash a následně budou kombinovány se skripty psanými v Pythonu. Jedná se o skripty pro vytvoření záznamů trénovací množiny dat sloužících k natrénování kaskády. Dále se jedná o skripty pro otestování úspěšnosti detekce objektu a srovnání samotných metod.

V následujících sekcích bude přiblížena implementace metod pro detekci objektů, resp. plechovek, na hrací ploše. Dále zde budou popsány některé skripty, které byly použity při trénování kaskády či při testování metod.

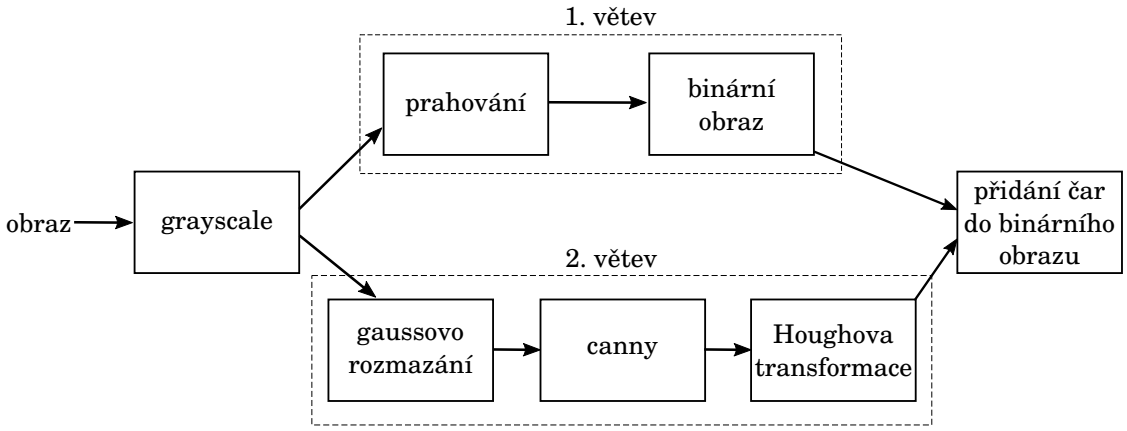
Při implementaci je vhodné využívat předimplementované metody z uvedených knihoven, protože pracujeme s obrazem ve vysokém rozlišení 2592×1944 pixelů. Metody v těchto knihovnách jsou psány v nižších programovacích jazycích (např. C) a jsou velmi dobře optimalizovány. Z tohoto důvodu jsou schopny efektivně vyřešit řadu problémů, se kterými se při zpracování obrazu setkáváme. Vysoké rozlišení obrazu bylo zvoleno převážně proto, abychom byli schopni co možná nejpřesněji z obrazu odhadnout dodatečné informace o vyhledaném objektu. Takovou informací by mělo být přibližné určení vzdálenosti objektu od kamery a určení úhlu, o který je robot vzhledem k objektu pootočen. Jedná se pouze o přibližné určení kvůli zkreslení informace v pořízeném obraze.

5.1 Metoda založená na Houghově transformaci

V této sekci bude probrána první metoda implementující počítačové vidění pro robota. Jak již bylo v předešlých kapitolách nastíněno, tato metoda je založena na vhodné segmentaci obrazu, která byla navržena na základě hrací plochy ve tvaru sítě. Informace o objektech na hrací ploše budou získávány na základě jediného snímku pořízeného kamerou robota. Více informací o hrací ploše, robotovi a soutěži, které se má zúčastnit, je uvedeno v sekci Ketchup House.

Na Obrázku 17 je znázorněn diagram, který popisuje proces zpracování obrazu a následné získání přímek pomocí Houghovy transformace. Nyní budou detailněji popsány kroky uvedené ve výše zmíněném diagramu.

Na obrázku 21a je demonstrován původní snímek hrací plochy s plechovkami, které je nutné v co možná nejkratším čase detekovat a lokalizovat.



Obrázek 17: Diagram zachycuje předzpracování a dále aplikování metod určených k segmentaci obrazu. Nejprve je obraz převeden do grayscale. Obraz ve stupních šedi je pak dále zpracováván dvěma různými větvemi, jak je zobrazeno v diagramu. V první větvi je na něj aplikováno adaptivní prahování, čímž získáme binární obraz, který lze velmi rychle zpracovávat. Ve druhé větvi je naším cílem získat přímky, které maximálně odpovídají čárám hrací plochy. K tomu je žádoucí na obraz ve stupních šedi aplikovat Gaussovo rozmazání pro odstranění šumu v obraze. Následně je na obraz aplikován Cannyho hranový detektor pro zvýraznění hran a na závěr Houghova transformace, kterou získáme výše zmíněné přímky. Kvůli demonstraci zpracování obrazu pro uživatele převádím navíc binární obraz zpět do barevného modelu v RGB a následně v něm vyznačím vyhledané přímky. Všechny následující operace jsou však prováděny na původním binárním obraze.

Nejprve je obrázek převeden z barevného modelu RGB do obrazu ve stupních šedi (Grayscale). Zpracování obrazu bude nyní rozděleno do dvou samostatných větví, kde v každé větvi vycházíme z výše zmíněného obrazu. Každá větev bude samostatně popsána v následujících podsekcích.

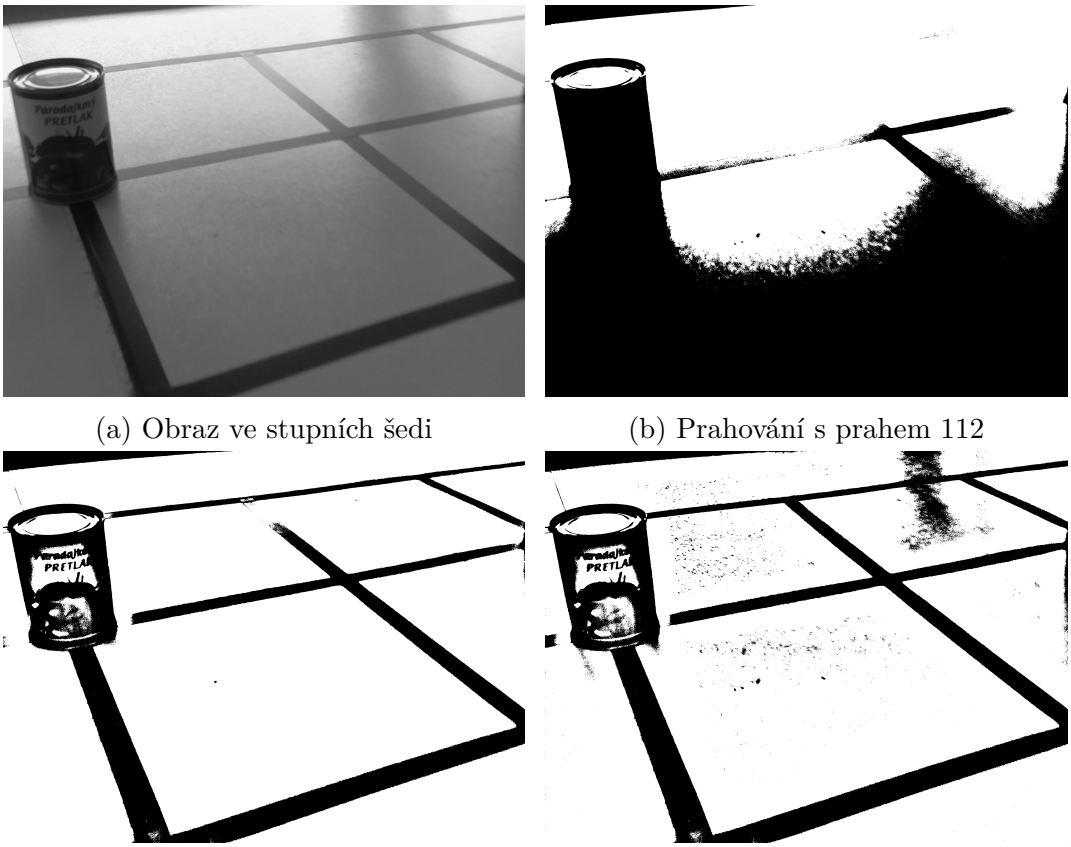
Detekování přímek - 1. větev

V prvním větvi je naším cílem získat binární obraz, nad kterým lze provádět velmi rychlé operace s použitím knihovny NumPy. Pro převedení obrazu ve stupních šedi do binárního obrazu lze použít prahování. Detailnější popis této metody je uveden v sekci Prahování.

V prvotní implementaci byla využita standardní metoda z knihovny OpenCV `threshold` s prahovou hodnotou 112, která podávala vynikající výsledky na původní testovací množině dat. Množina dat byla sestavena zhruba z 50 snímků, které byly pořízeny v jednom dni za určitých světelných podmínek. Po pořízení větší množiny testovacích dat, která se skládala zhruba z 1000 snímků, bylo zjištěno, že se nejedná o ideální metodu pro vytvoření binárního obrazu. Za dobu, než byla pořízena tak rozsáhlá množina testovacích dat, se v místnosti vystřídalo mnoho intenzit světla.

To vedlo k získání přesvětlených, anebo naopak velmi tmavých fotek. Po provedení prahování byla z obrazu vytracena téměř veškerá informace o hracím poli a samozřejmě i o hledaných plechovkách.

Řešením tohoto problému je použití adaptivního prahování, které je detailněji popsáno v podsekci Adaptivní prahování. Adaptivní prahování je ideálním řešením tohoto problému, jelikož počítá prahovou hodnotu pro každý pixel v obraze zvlášť na základě okolních pixelů. Na Obrázcích 18 a 19 jsou demonstrovány dva problémové snímky, které jsou v horní části přesvětleny a ve zbylých částech obrazu jsou velmi tmavé.

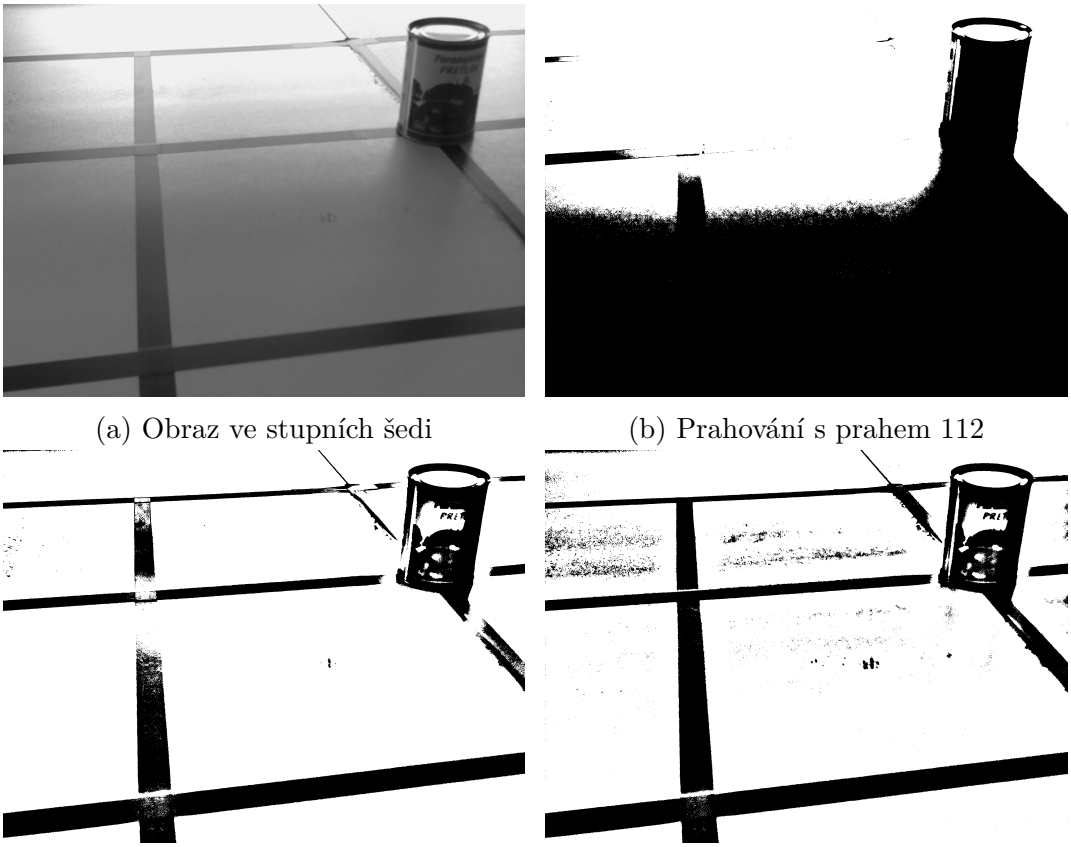


(a) Obraz ve stupních šedi (b) Prahování s prahem 112

(c) Adaptivní prahování - GAUSSIAN (d) Adaptivní prahování - MEAN

Obrázek 18: Na Obrázcích jsou demonstrovány různé typy prahování. Při použití (globálního) prahování je jasně viditelná ztráta informace v horní části obrazu, kde zmizely některé čáry hrací plochy, a stejně i v dolní části obrazu, která je kompletně začerněna. Obě implementace adaptivního prahování podaly velmi dobré výsledky. Z obrázků je však zřejmé, že nejlepší výsledky zde získáme při použití `ADAPTIVE_THRESH_GAUSSIAN_C`, který je schopen z obrazu odstranit veškerý nežádoucí šum.

Účinnost adaptivního prahování je závislá na velikosti zpracovávaného okolí pixelů. Např. v Obrázku 19d určuje prahovou hodnotou z okolí pixelu o velikosti



(a) Obraz ve stupních šedi

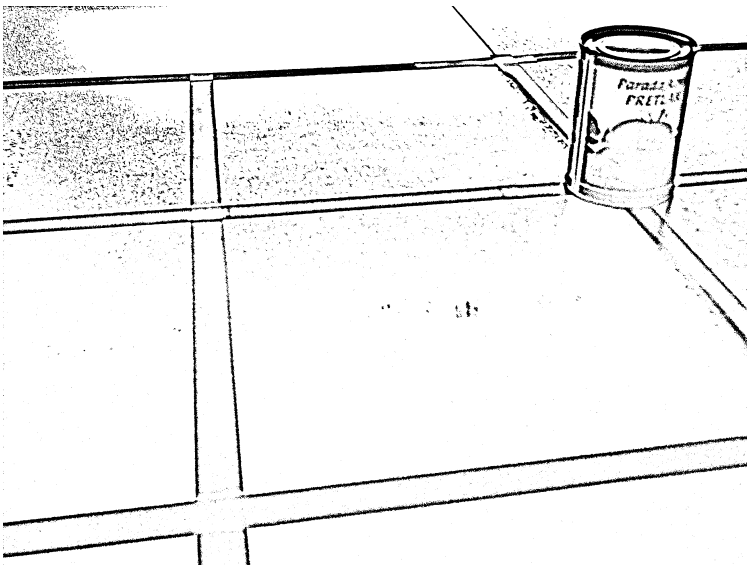
(b) Prahování s prahem 112

(c) Adaptivní prahování - GAUSSIAN

(d) Adaptivní prahování - MEAN

Obrázek 19: Na obrázcích jsou demonstrovány různé typy prahování. Při použití (globálního) prahování je téměř nemožné v obraze rozeznat, kde původně vedla čára hrací plochy. Více než polovina obrazu je kompletně začerněna. Adaptivní prahování podalo v obou případech velmi dobré výsledky. Zde však není zcela zřejmé, která implementace podala nejlepší výsledek. Adaptivní prahování s použitím `ADAPTIVE_THRESH_GAUSSIAN_C` opět z obrazu odstranil nežádoucí šum, ale nebyl schopen zcela zvýraznit čáry hrací plochy. Při použití `ADAPTIVE_THRESH_MEAN_C` jsou čáry hrací plochy zvýrazněny lépe. V obraze se místy vyskytuje šum, ale pouze v zanedbatelné míře, detekce by tedy neměla být ovlivněna.

231 pixelů. Je to dáno nadměrnou velikostí zpracovávaného obrazu s rozlišením 2592×1944. Ukázka adaptivního prahování se získáním prahové hodnoty z menšího okolí je zobrazena na Obrázku 20. Díky nedostatečně velkému okolí pixelů dojde k odstranění výplně čar hrací plochy i plechovky.



Obrázek 20: Adaptivní prahovaní s prahovací hodnotou určenou z okolí pixelu o velikosti 31 pixelů.

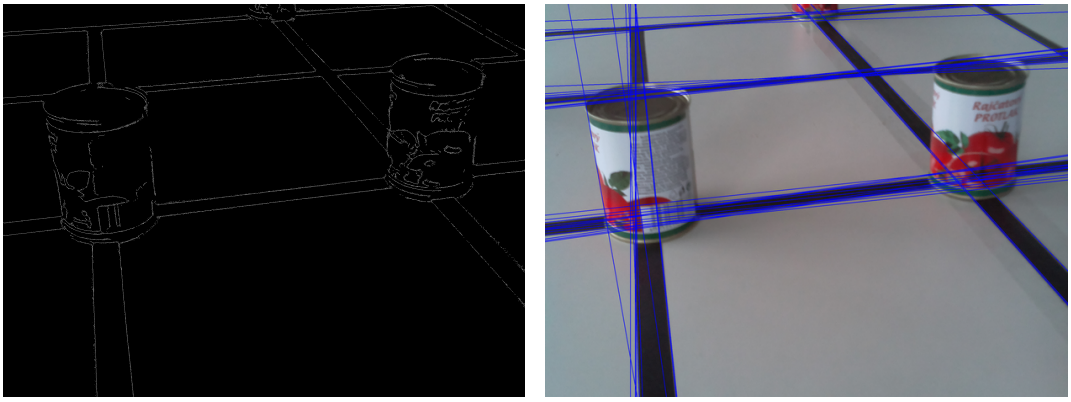
Detekování přímek - 2. větev

V této větvi se snažíme upravit obraz takovým způsobem, abychom co nejpřesněji získali přímky odpovídající poloze čar hrací plochy. Nejprve je obraz vyhlazen pomocí Gaussova rozmazání, abychom potlačili nežádoucí šum v obraze. Při zpracování byly nastaveny rozměry Gaussova jádra na hodnotu 5×5. Obraz po aplikaci Gaussova rozmazání je demonstrován na Obrázku 21b. Více informací o Gaussově rozmazání je uvedeno v sekci Gaussian Blur. Dále je nutné v takto upraveném obraze

(a) Původní obrázek (b) Vyhlazený obrázek ve stupních šedi

Obrázek 21: Na obrázcích je demonstrováno prvotní předzpracování v podobě převodu barevného modelu z RGB do obrazu ve stupních šedi. Zároveň zde již bylo aplikováno i Gaussovo rozmazání.

zvýraznit hrany. Na základě analýzy metod, kterou provedli Akram a Ismail (2013) v článku *Comparison of Edge Detectors*, byla vybrán Cannyho hranový detektor, jelikož poskytuje velmi dobré výsledky detekce hran a to i v obraze, ve kterém je přítomen šum. Podrobnější popis této metody je uveden v sekci Cannyho hranový detektor. Konkrétně v naší metodě byl výše zmíněný detektor aplikován s parametry 12 pro první prahování a 22 pro druhé prahování. Výsledek je zobrazen na Obrázku 22a.



(a) Zvýraznění hran metodou Canny (b) Vykreslení přímek do obrazu

Obrázek 22: Na Obrázku 22a jsou zvýrazněny hrany po aplikování Cannyho hranového detektoru. Na Obrázku 22b jsou zvýrazněny přímky, které jsme získali aplikací Houghovy transformace na Obraz 22a s detekovanými hranami.

Obraz se zvýrazněnými hranami je vhodný pro aplikování Houghovy transformace přímek. Sekce Houghovy transformace podrobně definuje použitou metodu pro vyhledání přímek v obraze. Na Obrázku 22b je zobrazen původní snímek, do kterého byly vykresleny přímky získané výše zmíněnou transformací.

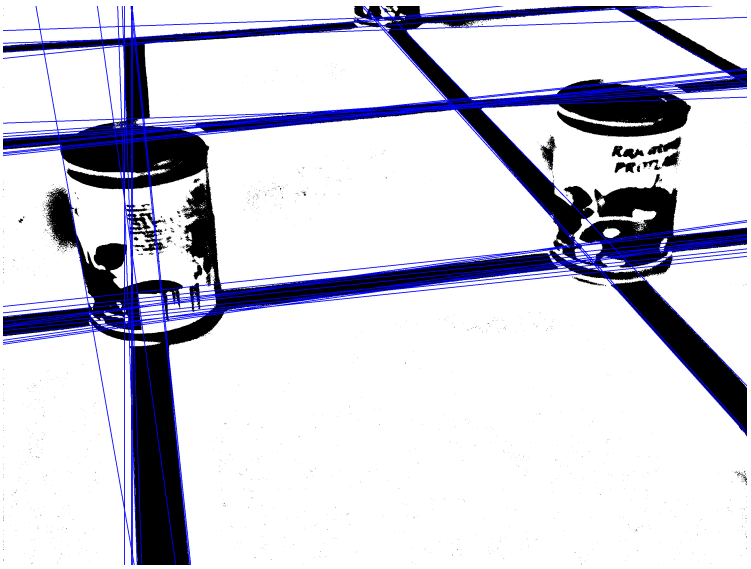
Detekování přímek

Nyní máme obraz převedený do binární podoby a zároveň jsme detekovali přímky v obraze. Obraz v binární podobě bude nyní využíván v kombinaci s metodami z knihovny OpenCV a NumPy za účelem dosažení velmi rychlého zpracování obrazu.

Zároveň si binární obraz převedeme i zpět do barevného modelu RGB, abychom zde mohli vykreslit detekované přímky. Tahle operace se provádí za účelem demonstrace zpracování obrazu uživateli a k debuggování metody, resp. nastavení vhodných parametrů, pro dosažení co možná nejlepších výsledků detekce plechovek. Takto upravený obraz je demonstrován na Obrázku 23.

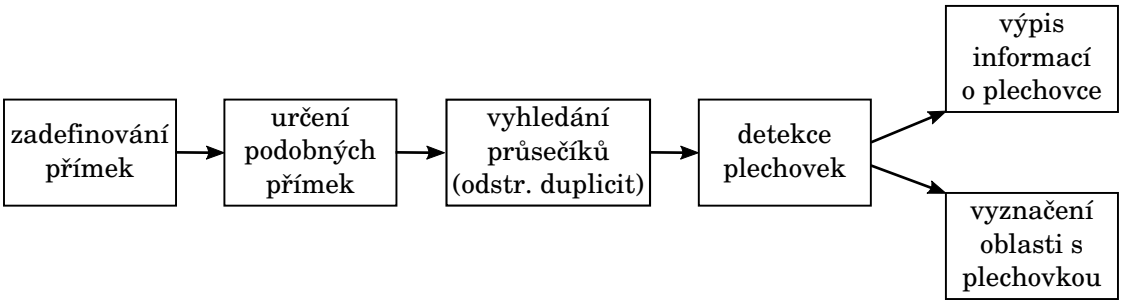
Zpracování přímek

Po úspěšném předzpracování obrazu a detekci přímek je nutné vyhledat plechovky. Za tímto účelem je žádoucí provést několik dalších operací právě na výše zmíněných



Obrázek 23: Binární obraz převedený zpět do barevného modelu RGB pro vykreslení přímek za účelem debuggování.

přímkách. Proces následujícího zpracování je popsán v diagramu na Obrázku 24.



Obrázek 24: Diagram plynule navazuje na předchozí diagram uvedený v obraze 17, kde jsme získali binární obraz a obraz se zvýrazněnými přímkami. Přímky jsou dále blíže definovány ve třídách z důvodu lepší přehlednosti kódu a debuggingu. Následně jsou přímky s podobnými vlastnostmi seskupeny do tříd. Mezi jednotlivými třídami přímek jsou hledány průsečíky. Dále probíhá odstranění duplicit (průsečíků, které jsou příliš blízko u sebe). V místech průsečíků jsou hledány potenciální plechovky. Na závěr se provádí výpis informací o detekovaných plechovkách na standardní výstup, anebo je přímo jejich pozice vyznačena do obrazu.

Nejprve je nutné přímky zadefinovat pomocí třídy pro pohodlnou práci s nimi. Každá z přímek byla popsána záznamem, který zahrnuje souřadnice počátečního bodu, koncového bodu, vektoru a unikátního identifikátoru. Jedním z důvodů, proč ukládat jak počáteční, tak koncový bod, je lepší přehlednost při odstraňování chyb v metodě. Unikátní identifikátor je následně využíván ke kontrole, jestli daná přím-

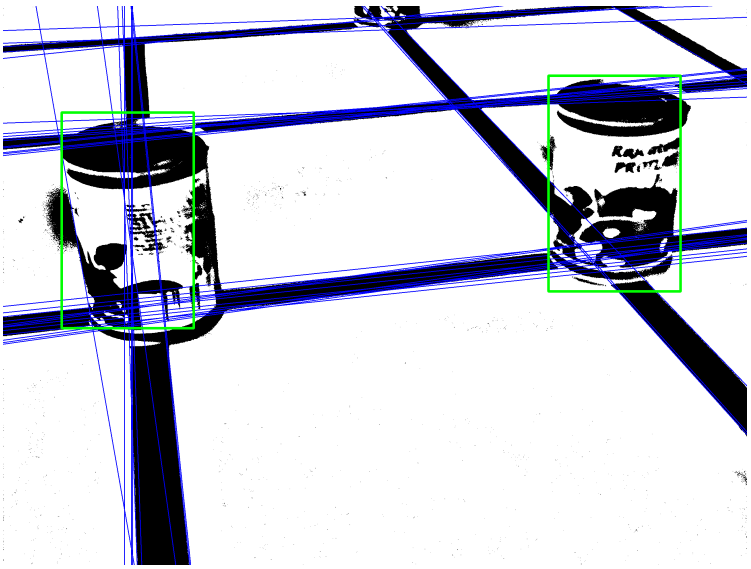
ka již byla zpracována, aby bylo zamezeno případným duplicitám či nadbytečným operacím.

Jakmile máme přímky zadefinovány, je důležité je roztřídit do skupin s podobnými vlastnostmi. Jedná se především o polohu přímky v obraze. Jakmile jsou přímky roztříděny do skupin, je nutné získat množinu průsečíků přímek z různých skupin. Následně omezíme tuto množinu o takové průsečíky, které jsou příliš blízko u sebe, protože je nevhodné v jedné oblasti opakovaně vyhledávat plechovku.

Na výsledné množině průsečíků se pokusíme vyhledat výše zmíněnou plechovku. Na základě umístění průsečíku v obraze bude vybrána velikost oblasti zájmu (ROI), ve které očekáváme plechovku. V této oblasti bude vypočítán poměr bílých a černých pixelů pomocí metody `countNonZero` z knihovny OpenCV, který stanoví, jestli se v oblasti vyskytuje nějaký objekt či nikoliv.

O vyhledané plechovce si ukládáme souřadnice původního průsečíku, kde byla plechovka vyhledávána, souřadnice bodu spodní a horní části stanovené oblasti zájmu, výšku objektu a příznak, jestli se jednalo o nějaký objekt. Spodní a horní část oblasti zájmu je dále zpřesňována kvůli dodatečným operacím, jako je odhad vzdálenosti objektu od kamery. Při zpřesňování této hodnoty se na y souřadnici hledá první pixel, který náleží objektu. Často se v obraze stává, že obsahuje po vyprahování nežádoucí šum a metoda není schopna zpřesnit výše zmíněnou oblast.

Na závěr se vypíší informace o vyhledaných objektech. Metoda předpokládá, že všechny vyhledané objekty jsou plechovkami. Na základě zvolených přepínačů při spuštění metody dojde k výpisu všech získaných informací, anebo pouze zkrácené verze ve formě vzdálenosti objektu od kamery a úhlu, o který je robot vzhledem k objektu pootočen. Tyto dvě hodnoty jsou určeny pro robota. Rovněž lze při spuštění specifikovat výstupní soubor, přičemž dojde ke zvýraznění vyhledaného objektu v obraze a následnému zápisu do výstupního souboru. Tato funkce slouží převážně pro odstranění chyb a maximální vyladění metody. Demonstrace tohoto výstupu je zobrazena na Obrázku 25.



Obrázek 25: Zvýraznění vyhledaných objektů v obraze za účelem odstraňování chyb a celkovému vyladění metody.

5.2 Metoda založená na strojovém učení

V této sekci bude popsána druhá metoda poskytující počítačové vidění pro robota. Tato metoda bude založena na strojovém učením. Jak již bylo dříve nastíněno v předchozích sekcích, metoda je závislá na důsledném natrénovaní kaskády slabých klasifikátorů a samozřejmě na výběru vhodného typu příznaků. Na základě průzkumu analýz provedeného v kapitole Metodika práce byly k natrénování kaskády slabých klasifikátorů vybrány příznaky LBP a Haar.

Pokud bychom chtěli detekovat objekt typu tvář, oko, nos nebo ústa, bylo by možné využít jednu z již natrénovaných kaskád klasifikátorů, které jsou volně přístupné na internetu. Jelikož se v našem případě jedná o netradiční objekt v podobě plechovky od kečupu, nezbývá nám nic jiného než natrénovat novou kaskádu klasifikátorů.

V následujících sekcích bude popsán proces trénování kaskády klasifikátorů, dále zde bude popsána výsledná metoda využívající natrénovanou kaskádu a uživatelské rozhraní této metody.

Natrénování kaskády klasifikátorů - základ

Základem pro natrénování kaskády klasifikátorů je mít na svém operačním systému nainstalovanou knihovnu OpenCV a předpřipravený adresář s vhodně pojmenovanými soubory obsahující informace o trénovaném objektu a složkami, kde se budou vyskytovat trénovací data a kam budou ukládány výstupní soubory.

Dále je nutné získat trénovací vzorky. Trénovací vzorky jsou tvořeny z pozitivních a negativních snímků.

Pozitivní a negativní snímky

Pozitivní snímky jsou takové snímky, ve kterých se vyskytuje objekt, jenž chceme detekovat. Na negativních snímcích se tento objekt nesmí v žádném případě vyskytovat. Negativní snímky by měly zahrnovat i okolí, ve kterém se náš hledaný objekt může objevit (např. v podobě hrací plochy). Dále by v těchto snímcích neměli scházet objekty, které jsou tvarem, barvou či jiným způsobem podobné našemu hledanému objektu, za účelem zpřesnění výsledné detekce.

Snímky, na kterých se vyskytuje objekt, jenž bychom chtěli detekovat, pravděpodobně obsahují i dodatečné informace o nějakém okolí, kde byl snímek pořízen. Tato dodatečná informace je pro trénování kaskády klasifikátorů nežádoucí. Pořízené pozitivní snímky je tedy nutné nejprve předpřipravit. Jak vybrat ve snímku pouze hledaný objekt, je uvedeno následovně:

- Oříznutím obrázku v místech, kde se vyskytuje hledaný objekt.
- Vytvořením záznamu s informacemi o tom, kde se v daném obraze vyskytují hledané objekty a v jakém počtu.

Obě tyto možnosti jsou časově náročné. Z tohoto důvodu byla zvolena druhá možnost, kterou lze částečně urychlit, a to pomocí skriptu.

Skript na obrazovku jednotlivě předkládá obrázky a uživatel v nich pomocí zaklikávání myší definuje pozice hledaných objektů. Všechny informace se uloží v požadovaném formátu do externího souboru.

Počet trénovacích vzorků závisí na mnoha faktorech jako např. na kvalitě obrazu, typu objektu, výkonu procesoru atd. Všeobecně platí, že velikost sady negativních snímků by měla značně převyšovat velikost sady pozitivních snímků. Spolehlivý klasifikátor jsme získali až po natrénování kaskády z více jak tisíce pozitivních snímků a přibližně trojnásobné množiny negativních snímků.

Vytváření vzorků

U negativních snímků není nutné specifikovat žádné bližší informace. Pouze je nutné vytvořit textový soubor, ve kterém bude na každém řádku specifikována cesta k negativnímu snímku. Uživatelé, kteří používají operační systém linux, mohou využít nástroj `find`. Vzniklý soubor nám bude sloužit jako seznam negativních vzorků.

U pozitivních snímků není převod na vzorky tak jednoduchý jako v případě negativních snímků. Existuje více způsobů, jak tohoto převodu docílit. V našem případě byl využit nástroj `opencv_createsamples`, který na vstupní data v podobě pozitivních snímků uplatní operace transformace a distorze. Výstup z této funkce je uložen do souboru s příponou `vec`, který již lze využít k natrénování kaskády klasifikátorů.

Natrénování kaskády klasifikátorů

Knihovna OpenCV nabízí dvě různé metody pro natrénování kaskády klasifikátorů a to `opencv_haartraining` a `opencv_traincascade`. Při trénování klasifikátoru byla zvolena metoda `opencv_traincascade`, protože je nejen novější, ale především podporuje oba příznaky LBP i Haar (Bradski, 2000).

Při použití této metody je nutné vhodně nastavit značné množství parametrů. Význam těchto parametrů lze dohledat v dokumentaci OpenCV. Z tohoto důvodu zde zmíním pouze několik zásadních parametrů, aby bylo možné zreprodukovat trénink klasifikátoru obdobným způsobem. Parametr `-numstages` ovlivňuje počet stupňů (stages) kaskády, které budou natrénovány. V mé kaskádě slabých klasifikátorů bylo natrénováno 28 stupňů s příznaky LBP a 17 stupňů s příznaky Haar. Parametr `-minHitRate` specifikuje požadovanou přesnost (hit rate) zvlášť pro každý stupeň tréninku klasifikátoru. Tento parametr byl nastaven na hodnotu 0.99. Parametry `-w` a `-h` definují rozměry vzorku (80 a 105 pro LBP a 27 a 32 pro Haar). Parametry `-precaalcValBufSize` a `-precalcldxBufSize` byly nastaveny na hodnotu 1024. Tyto parametry definují velikost bufferu pro přípravné výpočty (precalculated) příznaků v jednotkách Mb. Čím větší buffer bude nastaven, tím rychleji proběhne trénink (Bradski, 2000).

Proces učení může zabrat velmi dlouhou dobu, řádově se pohybuje v několika dnech až týdnech. Jak již bylo dříve naznačeno, vliv na křivku doby učení má mnoho faktorů jako jsou např. dostupný výpočetní výkon, počet trénovacích stupňů, stanovená přesnost jednoho stupně a mnoho dalších.

Doba učení každého dalšího stupně roste velmi strmě. Pro ilustraci lze uvést, že trénink několika prvních stupňů proběhne za nějakou dobu a při trénování dalšího jediného stupně může být tato celková doba až zdvojnásobena.

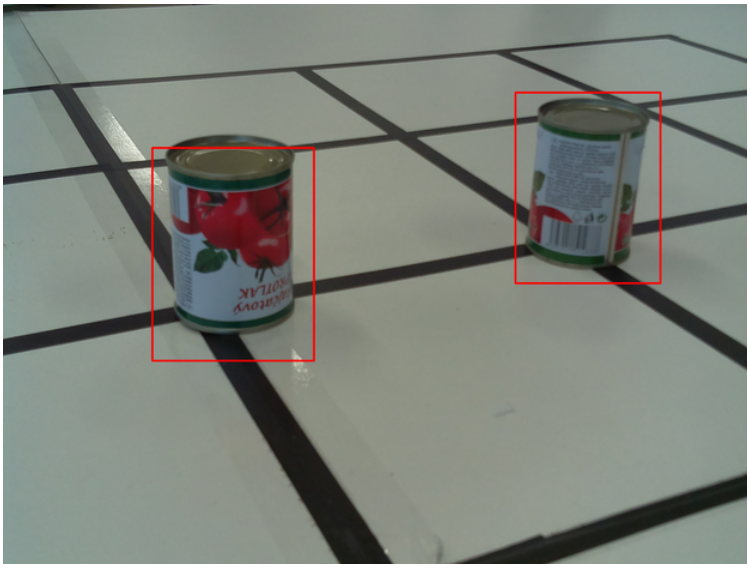
Metoda byla navržena tak, aby každý stupeň sloužil jako záchytný bod a nebylo nutné začínat učení při případném narušení (např. pád kernelu) pokaždé od začátku. Jedná se však nejen o zdlouhavý, ale i výpočetně náročný proces. Z tohoto důvodu při běhu programu pravděpodobně nebude možné využívat danou stanici.

Výsledná metoda

Implementace výsledné metody poskytující počítačové vidění pro robota na základě natrénované kaskády bude uskutečněna v jazyce Python, aby srovnání obou metod proběhlo v co možná nejvíce rovnocenných podmínkách.

Metoda funguje následujícím způsobem: Na vstupu obdrží snímek, na kterém se vyskytuje hrací pole s plechovkami. Obrázek je následně převeden do barevného modelu ve stupních šedi a pomocí natrénované kaskády klasifikátorů jsou vyhledány všechny plechovky v obraze (Obrázek 26).

Na závěr je nutné získaná data o plechovkách převést do jednotné formy, která je shodná s výstupem z první metody. Jednotný výstup je vyžadován kvůli testování a zároveň bude využit jako vstupní parametr pro metodu, jež slouží k odhadu úhlu,



Obrázek 26: Detekované plechovky pomocí kaskády klasifikátorů, konkrétně v tomto případě byly využity příznaky LBP.

o který je robot vzhledem k plechovce pootočen a rovněž k odhadu vzdálenosti kamery od objektu.

5.3 Uživatelské rozhraní

Uživatelské rozhraní bylo navrženo jednotně pro obě dvě metody z důvodu snazšího testování, a to formou příkazové řádky.
Použití:
`can_detection.py | can_detect_line.py [-h] [--long_output] [--testing_output TESTING_OUTPUT] [--input INPUT] [--output OUTPUT]`
Přehled parametrů:

- `-h, --help` - Výpis nápovědy.
- `--long_output, -l`: Nepovinný parametr zajišťující výpis detailní zprávy o vyhledaných objektech.
- `--testing_output TESTING_OUTPUT, -t TESTING_OUTPUT`: Nepovinný parametr, který zajišťuje výpis informací o plechovkách v požadovaném tvaru pro testování.
- `--input INPUT, -i INPUT`: Povinný parametr pro určení cesty k vstupnímu souboru (snímek hrací plochy).
- `--output OUTPUT, -o OUTPUT`: Nepovinný parametr určující umístění výstupního souboru. Bude se jednat o snímek, ve kterém jsou prostřednictvím oblasti zájmu vyznačeny plechovky.

6 Určení polohy objektu v prostoru

Cílem práce je co nejrychleji s co možná největší přesností detekovat hledaný objekt v obraze. Po úspěšné detekci objektu se pokusíme přesněji určit jeho polohu v reálném prostoru pomocí dvojice hodnot. Jmenovitě se jedná o vzdálenost objektu od kamery a o úhel pootočení robota vzhledem k objektu.

Než bude možné detekovat vzdálenost objektu od kamery, je nejprve nutné určit ohniskovou vzdálenost kamery.

6.1 Ohnisková vzdálenost kamery

Ohnisková vzdálenost kamery, kterou robot disponuje, je dle dokumentace $(3.60 \pm 0.01)\,mm$ (Raspberry Pi, 2016). Danou hodnotu je nutné přepočítat z důvodu umístění kamery na horní části robota. Tato kamera je navíc sklopena o určitý úhel, aby zabírala co možná největší část hrací plochy. Výpočet ohniskové vzdálenosti bude mírně zkreslený vinou naklopení kamery. Aby toto zkreslení nebylo příliš výrazné, je nutné při výpočtu použít skutečnou výšku plechovky, kterou snímá naklopená kamera. Přestože výpočet kvůli negativním vlivům, jako je naklopení kamery a detekce výšky plechovky v pixelech pomocí naimplementovaných metod, nebude zcela přesný, bylo by užitečné získat alespoň přibližnou informaci o vzdálenosti robota od detekovaného objektu.

Při výpočtu ohniskové vzdálenosti bylo vycházeno z následující rovnice:

$$fl = \frac{dto \cdot oh \cdot sh}{rh \cdot ih}, \tag{13}$$

kde dto je vzdálenost od objektu v mm,
fl je ohnisková vzdálenost v mm,
rh je reálná výška objektu v mm,
ih je výška obrazu v px,
oh je výška objektu v px,
sh je výška senzoru v mm.

Tabulka 1 znázorňuje výpočet ohniskové vzdálenosti, které bylo provedeno před srovnáním přesnosti určení vzdálenosti kamery od objektu, a úhlu pootočení robota vzhledem k objektu.

Výsledná hodnota ohniskové vzdálenosti je $fl = (3.607 \pm 0.237)\,mm$ s relativní chybou $\rho = 6.564\,\%$.

6.2 Výpočet vzdálenosti objektu od kamery

Při výpočtu vzdálenosti objektu od kamery lze vycházet z rovnice 13 a vyjádřit proměnnou dto, jak je demonstrováno v rovnici 14.

$$dto = \frac{fl \cdot rh \cdot ih}{oh \cdot sh} \tag{14}$$

V předešlé podsekci Ohnisková vzdálenost kamery jsme měřením určili ohniskovou vzdálenost kamery fl. Rovnice již neobsahuje žádnou neznámou, která by bránila výpočtu vzdálenosti objektu od kamery. Tabulka 2 shrnuje výsledky testování úspěšnosti detekce vzdálenosti objektu od kamery.

Z naměřených hodnot je zřejmé, že se nejedná o vhodný způsob výpočtu přesné vzdálenosti kamery od objektu hned z několika důvodů. Nejprve je nutné zdůraznit, že detekce výšky objektu v pixelech pomocí metody je nepřesná. Jak bylo detailněji popsáno v kapitole Implementace metod, metoda využívá v oblasti, kde očekává plechovku oblast zájmu (ROI) s předdefinovanou výškou, která je následně zpřesňována, aby odpovídala výšce objektu v pixelech. Pokud se v obraze vyskytne nějaký negativní vliv (šum, čára hrací plochy), nedojde k výše zmíněnému zpřesnění. Z tohoto důvodu se v tabulce často vyskytují vzdálenosti se stejnou hodnotou. Oblast zájmu je vybírána tak, aby rozdíl od reálné hodnoty byl co možná nejnižší i v případě, že nedojde ke zpřesňování výšky objektu.

Pravděpodobně bychom získali přesnější výsledky, kdybychom kameru umístili maximálně do výšky plechovky, tedy 74 mm, a zabírali bychom tuto plechovku zpříma. Tím bychom ovšem zaznamenávali menší část hrací plochy a nezískali bychom z jednoho snímku takové množství informací, jaké získáme s výše zmíněným nastavením kamery. Odchylky při měření reálné vzdálenosti v milimetrech byly zanedbány z důvodu, že se metodou určená vzdálenost liší od reálné hodnoty řádově v jednotkách centimetrů.

6.3 Výpočet úhlu pootočení robota vzhledem k objektu

Výpočet úhlu, o který je nutné pootočit robota, aby byl následně nasměrován na plechovku, určíme pomocí polárních souřadnic, jak již bylo podrobněji popsáno v sekci Houghovy transformace. Podobně jako v minulé sekci bude výpočet zkreslený vlivem umístění kamery. Ideální poloha kamery by nyní byla v pozici, kdy by kamera zabírala hrací pole z ptačí perspektivy. Při výpočtu je dále nutné zohlednit, že kamera zabírá hrací plochu až 23 cm před robotem. Tabulka 3 shrnuje výsledky testování úspěšnosti detekce úhlu pootočení robota vzhledem k objektu.

Z průměrné hodnoty relativní odchylky $\overline{\delta} = 13.981\,\%$ lze usoudit, že se rovněž nejedná o vhodný způsob výpočtu úhlu pootočení robota vzhledem k objektu. K větším nepřesnostem často dochází na snímcích, kde se objekt vyskytuje dále od středu (na okraji snímku). To je dáno tím, že nebyly zjištěny přesné vnitřní parametry kamery. Tyto parametry by bylo možné získat kalibrací.

Odchylky ± 1 stupeň při měření reálné velikosti úhlu pootočení robota vzhledem k plechovce byly zanedbány z podobného důvodu jako v předcházející sekci.

č. měření	dto **[mm]**	oh **[px]**	fl **[mm]**	Δfl
1	480	422	3.547	0.004
2	300	578	3.036	0.326
3	580	360	3.656	0.002
4	350	534	3.272	0.112
5	400	487	3.411	0.039
6	450	453	3.569	0.001
7	500	426	3.729	0.015
8	550	392	3.775	0.028
9	600	335	3.519	0.008
10	380	520	3.460	0.022
11	440	438	3.374	0.054
12	270	690	3.262	0.119
13	300	651	3.419	0.035
14	330	603	3.484	0.015
15	360	585	3.687	0.006
16	390	540	3.687	0.006
17	420	513	3.772	0.027
18	450	482	3.798	0.036
19	470	465	3.827	0.048
20	500	444	3.887	0.078
21	530	420	3.897	0.084
22	560	408	4.000	0.155
23	260	693	3.155	0.205
24	290	649	3.295	0.097
25	320	627	3.513	0.009
26	350	590	3.616	0.001
27	380	558	3.713	0.011
28	410	514	3.690	0.007
29	440	495	3.813	0.043
30	460	474	3.818	0.044
31	490	441	3.783	0.031
32	520	435	3.961	0.125
			$\overline{fl} = 3.607\,mm$	
			$\mathrm{Var}(fl) = 0.056$	
			$\sigma = 0.237$	

Tabulka 1: Tabulka znázorňuje výpočet ohniskové vzdálenosti různě vzdálených plechovek. *dto* značí vzdálenost od objektu v milimetrech. *oh* vyjadřuje výšku objektu v pixelech. fl je hledaná ohnisková vzdálenost. $\overline{fl}$ značí její aritmetický průměr. Δfl je druhá mocniny odchylky fl a $\overline{fl}$. $\mathrm{Var}(fl)$ je rozptyl hodnot a σ zde znázorňuje směrodatnou odchylku.

č. měření	s **[mm]**	dto **[mm]**	δ **[%]**
1	270	327.465	21.283
2	300	327.466	9.155
3	330	355.846	7.832
4	360	355.846	1.154
5	390	355.846	8.757
6	420	437.515	4.170
7	450	437.515	2.774
8	470	437.515	6.912
9	500	534.115	6.823
10	530	510.612	3.658
11	560	588.468	5.084
12	260	266.884	2.648
13	290	266.884	7.971
14	320	355.846	11.202
15	350	355.846	1.670
16	380	355.846	6.356
17	410	437.516	6.711
18	440	437.516	0.565
19	460	437.516	4.888
20	490	437.516	10.711
21	520	495.203	4.769
22	550	560.794	1.963
			$\overline{\delta} = 6.230\,\%$

Tabulka 2: Tabulka znázorňuje srovnání detekované vzdálenosti objektu od kamery pomocí naimplementované metody s reálnou (naměřenou) vzdáleností. *dto* značí vzdálenost od objektu v milimetrech, která byla zjištěna naimplmenetovanou metodou. s je reálná naměřená vzdálenost. δ značí relativní odchylku a $\overline{\delta}$ její aritmetický průměr s hodnotou 6.230 %.

č. měření	**úhel získaný metodou**	**naměřený úhel**	δ [%]
1	9.659	13	25.700
2	6.785	6	13.083
3	20.871	19	9.847
4	21.720	23	5.565
5	3.492	3	16.400
6	17.687	15	17.913
7	9.278	8	15.975
8	0.751	1.5	49.933
9	17.308	18	3.844
10	21.892	18	21.622
11	8.040	6	34.000
12	13.750	13	5.769
13	11.171	11	1.555
14	10.766	11	2.127
15	13.682	14	2.271
16	11.423	12	4.808
17	11.888	13	8.554
18	21.763	18	20.906
19	5.995	5	19.900
20	3.223	4	19.425
21	19.889	16	24.306
22	7.946	8	0.675
23	18.136	15	20.907
24	10.505	13	19.192
25	15.022	14	7.300
26	13.734	13	5.646
27	19.889	18	10.494
28	11.551	12	3.742
			$\overline{\delta} = 13.981\,\%$

Tabulka 3: Tabulka znázorňuje srovnání detekce úhlu, o který je nutné pootočit robota, aby byl nasměrován na plechovku s reálnou hodnotou tohoto úhlu, která byla pořízena měřením. δ značí relativní odchylku a $\overline{\delta}$ její aritmetický průměr s hodnotou 13.981 %.

7 Srovnání metod

V této kapitole bude popsáno veškeré testování, které bylo aplikováno na implementované metody popsané v kapitole Implementace metod. Testováno bylo mnoho různých aspektů od doby běhu metody po stanovení přesnosti detekce metody.

Aby testování bylo co nejvíce prokazatelné, je nutné jej provést na velké množině testovacích dat. Tato data je navíc vhodné roztřídit do několika datasetů (skupin dat s podobnými vlastnostmi).

V následujících sekcích budou popsány jednotlivé metriky, které byly zvoleny pro testování metod, dále zde budou popsány jednotlivé datasety a na závěr zde budou uvedeny výsledky samotného testování.

7.1 Metriky srovnání metod

Tato sekce se věnuje vyhodnocování úspěšnosti detekce objektů. Jak již bylo dříve naznačeno, nejprve je nutné sestavit testovací sadu dat ve formě datasetů.

Ve snímcích testovací sady je nutné vyznačit oblasti, ve kterých se vyskytují námi hledané objekty. Tato oblast bývá označována jako ground truth. Tuto oblast definujeme v obraze pomocí souřadnic dvou bodů (levého horního a pravého dolního bodu), které získáváme prostřednictvím skriptu implementovaného v Pythonu. Tento skript je volán pro každý testovací snímek zvlášť v rámci shellovského skriptu, který postupně ukládá záznamy o počtu a umístění plechovek v daném snímku.

Pokud máme testovací sadu kompletní, lze přejít k samotnému testování. Při tomto testování dochází k porovnání ground truth s oblastí, která byla stanovena detektorem jako hledaný objekt. Pokud se tyto dvě oblasti překrývají z více jak 50 %, detekce byla úspěšná. V takovém případě je při testování označena jako skutečně pozitivní TP (true positive). V opačném případě se jedná o detekci falešně pozitivní FP (false positive). Rovněž se zaznamenává chybějící detekce plechovky v místě, kde se plechovka skutečně vyskytuje. Tato detekce se označuje jako falešně negativní FN (false negative) a v případě metody založené na Houghově transformaci budou zaznamenány i detekce, které jsou skutečně negativní TN (true negative).

Z takto definovaných veličin lze určit statistické informace popsané v následujících podsekcích.

Specificita

Specificita, neboli TNR (True negative rate), zde definujeme jako poměr správně vyloučených detekcí, kdy se skutečně nejednalo o plechovku vzhledem ke všem případům, kdy se nejednalo o plechovku. Výpočet je demonstrován v rovnici 15 (Vránová, Horák, Krátká, Hendrichová a Kovaříková, 2009).

$$specificita = TNR = \frac{TN}{FP + TN} \tag{15}$$

Senzitivita (recall)

Senzitivita, neboli recall či TPR (True positive rate), určuje podíl skutečně pozitivních detekcí vůči celkovému počtu plechovek v testovací sadě. Výpočet je demonstrován v rovnici 16 (Manning, Raghavan a Schütze, 2008).

$$senzitivita = recall = TPR = \frac{TP}{TP + FN} \tag{16}$$

Precision

Precision, neboli PPV (positive predictive value), nám udává míru pozitivních detekcí plechovek ze všech vybraných plechovek detektorem. Výpočet je uveden v rovnici 17.

$$precision = PPV = \frac{TP}{TP + FP} \tag{17}$$

F-measure

F-measure je váženým harmonickým průměrem hodnot precision a recall, které se vzájemně ovlivňují. Pokud dojde ke zvýšení hodnoty precision, sníží se tím hodnota recall a naopak. K nalezení ideálního poměru mezi těmito dvěma hodnotami slouží právě metrika F-measure. Výpočet je odvozen v rovnici 18 (Manning, Raghavan a Schütze, 2008).

$$F = \frac{1}{\alpha\frac{1}{precision} + (1-\alpha)\frac{1}{recall}} = \frac{(\beta^2 + 1)\ precision\ recall}{\beta^2\ precision + recall}, \tag{18}$$

kde $\alpha \in [0, 1]$ a $\beta \in [0, \infty]$. K tomu, abychom získali vyvážený (balanced) F-measure, který rovnoměrně váhuje presicion a recall, je třeba nastavit β na hodnotu 1. Vyvážený F-measure značíme jako $F1$. Výsledný výpočet je uveden v rovnici 19. (Manning, Raghavan a Schütze, 2008)

$$F1 = \frac{2\ precision\ recall}{precision + recall} \tag{19}$$

Přesnost

Přesnost (accuracy) zde definujeme jako podíl všech správně klasifikovaných oblastí zájmu vůči všem testovaným oblastem. Výpočet je uveden v rovnici 20 (Vránová, Horák, Krátká, Hendrichová a Kovaříková, 2009).

$$přesnost = \frac{TP + TN}{TP + FP + FN + TN} \tag{20}$$

7.2 Datasety

V této sekci budou detailněji popsány testované datasety. Jak již bylo zmíněno v předchozích sekcích, jedná se o seskupená data, které sdílejí podobné vlastnosti. V našem případě vytváříme datasety ze snímků, které byly pořízeny robotem.

První dataset bude obsahovat zaostřené snímky, kde se vyskytuje kompletní plechovka. To znamená, že bude obsahovat ideálně pořízené snímky, ve kterých by nemělo být tak náročné detekovat hledaný objekt. Ukázka takového snímku je zobrazena na Obrázku 27.

Obrázek 27: Zástupce datasetu obsahujícího zaostřené snímky.

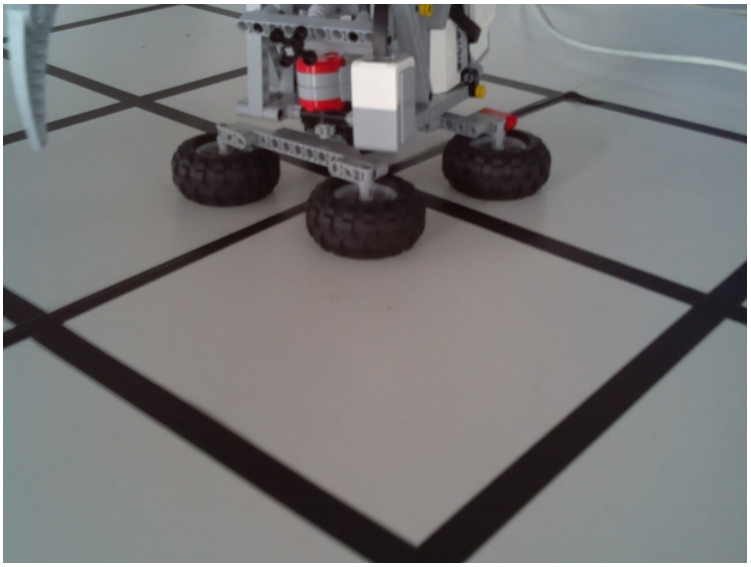
Druhý dataset bude zahrnovat snímky, které byly pořízeny robotem při pohybu (ve vyšších rychlostech) a tudíž jsou více či méně rozmazány. Ukázka takového snímku je zobrazena na Obrázku 28.

Třetí dataset bude obsahovat snímky, kde se nevyskytuje námi hledaný objekt, ale objekt zcela neznámý. Tento dataset má otestovat, zdali je metoda schopná odlišit námi hledaný objekt od cizího objektu, kterým může být např. konkurenční robot. Ukázka takového snímku je zobrazena na Obrázku 29.

Čtvrtý dataset obsahuje snímky, kde je zachycena pouze část plechovky. Bylo by užitečné odhadnout, jak velká část plechovky musí být v obraze viditelná, aby ji metoda byla schopná ve většině případů detekovat. Ukázka takového snímku je zobrazena na Obrázku 30.

Poslední dataset zahrnuje snímky, na kterých je zobrazena pouze hrací plocha bez jediného objektu. Tento dataset má otestovat, jestli hrozí, že některá část hrací plochy může být např. vlivem nekvalitního osvětlení detekována jako plechovka. Ukázka takového snímku je zobrazena na Obrázku 31.

Obrázek 28: Zástupce datasetu skládajícího se z rozmazaných snímků.

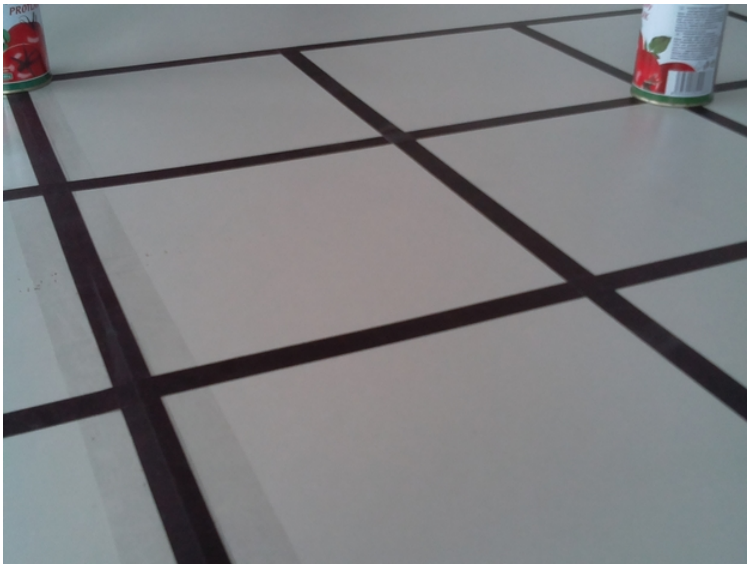


Obrázek 29: Zástupce datasetu skládajícího se ze snímků, kde se nevyskytuje náš hledaný objekt, ale naopak objekt cizí.

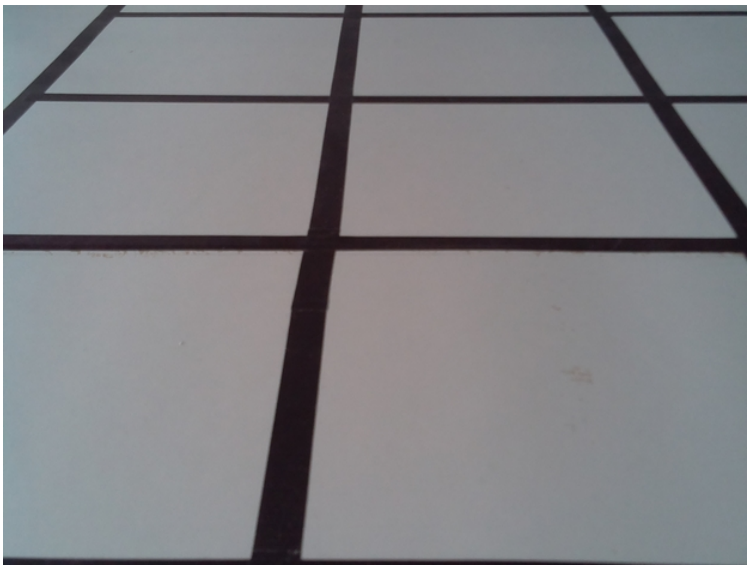
7.3 Nastavení parametrů

Tato sekce se věnuje vhodnému nastavení parametrů u obou implementovaných metod. Takové nastavení je možné stanovit na základě měření přesnosti detekce. Přesnost detekce se porovnává na množině všech platných hodnot sledovaného parametru. Zpřesňování metod bylo provedeno na datasetu, který byl složen ze snímků z každého datasetu popsaného v sekci Datasety.

Při hledání ideálního stupně natrénované kaskády slabých klasifikátorů s využi-



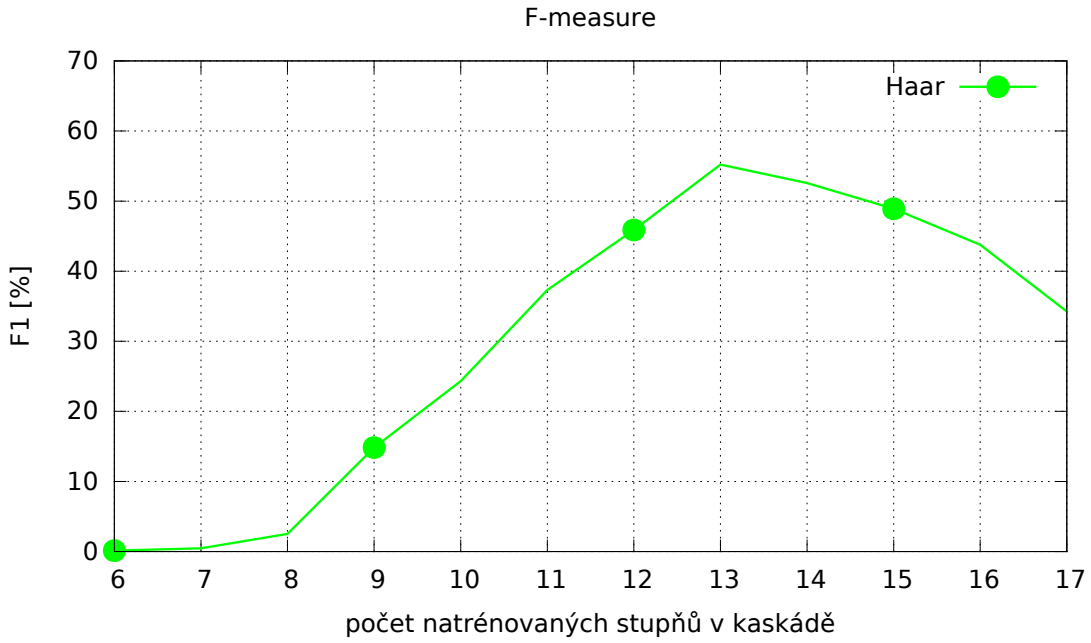
Obrázek 30: Zástupce datasetu tvořeného ze snímků, kde je zachycena pouze část plechovky.



Obrázek 31: Zástupce datasetu sestávajícího ze snímků, kde se nevyskytuje žádný objekt.

tím Haarových příznaků bylo zjištěno, že zdaleka nedosahuje tak dobrých výsledků jako při použití LBP příznaků. Navíc Haarova kaskáda slabých klasifikátorů se dopouštěla více chybných detekcí. Úspěšnost detekce jednotlivých stupňů natrénovaných kaskád s využitím příznaků Haar je zobrazena v Obrázku 32. Jelikož kaskáda slabých klasifikátorů s LBP příznaky poskytla ve všech ohledech lepší výsledky než v případě využití příznaků Haar a doba zpracování je v obou případech srovnatelná,

byla ke srovnání s metodou založenou na Houghově transformaci vybrána **kaskáda slabých klasifikátorů s LBP příznaky**.



Obrázek 32: V grafu byly jednotlivé kaskády, s určitým počtem natrénovaných stupňů, ohodnoceny metrikou F-measure. Nejvíce byla ohodnocena kaskáda s natrénovanými 13 stupněmi.

Metoda založená na Houghově transformaci

V této podsekci se pokusím o zpřesnění detekce nalezením vhodné hodnoty prahu. Tabulka 4 zobrazuje dosavadní přesnost metody, kterou bychom chtěli navýšit.

práh	**TP**	**FN**	**FP**	**TN**	**přesnost** [%]	**F1** [%]
55	108	733	17	45707	98.389	22.360

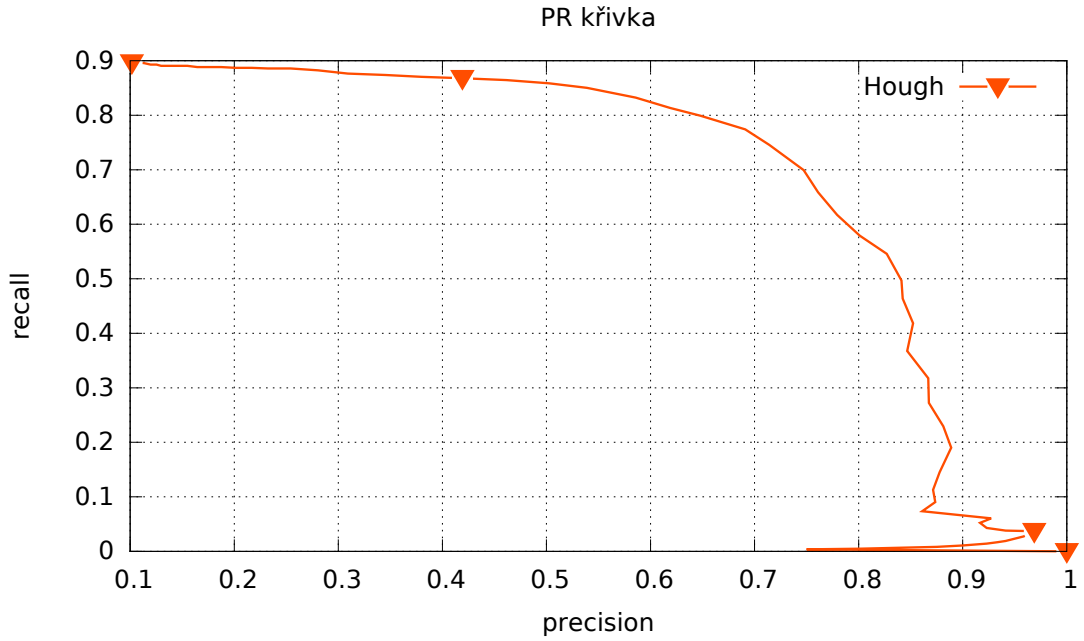
Tabulka 4: Tabulka znázorňuje původní přesnost metody (před vyhledáním vhodných parametrů).

Úspěch detekce metody založené na Houghově transformaci závisí na vhodném nastavení poměru černých pixelů vůči bílým pixelům v testované oblasti, kde je očekávána přítomnost plechovky.

Pokud černé pixely v oblasti zabírají procentuální část plochy vyšší, než je stanovený práh, znamená to, že se pravděpodobně jedná o plechovku. Při hledání vhodné hodnoty prahu byl testován celý rozsah [1-99 %]. V grafu na Obrázku 33 je vykreslena PR křivka. Ideální hodnoty se pohybují v levé horní části grafu.

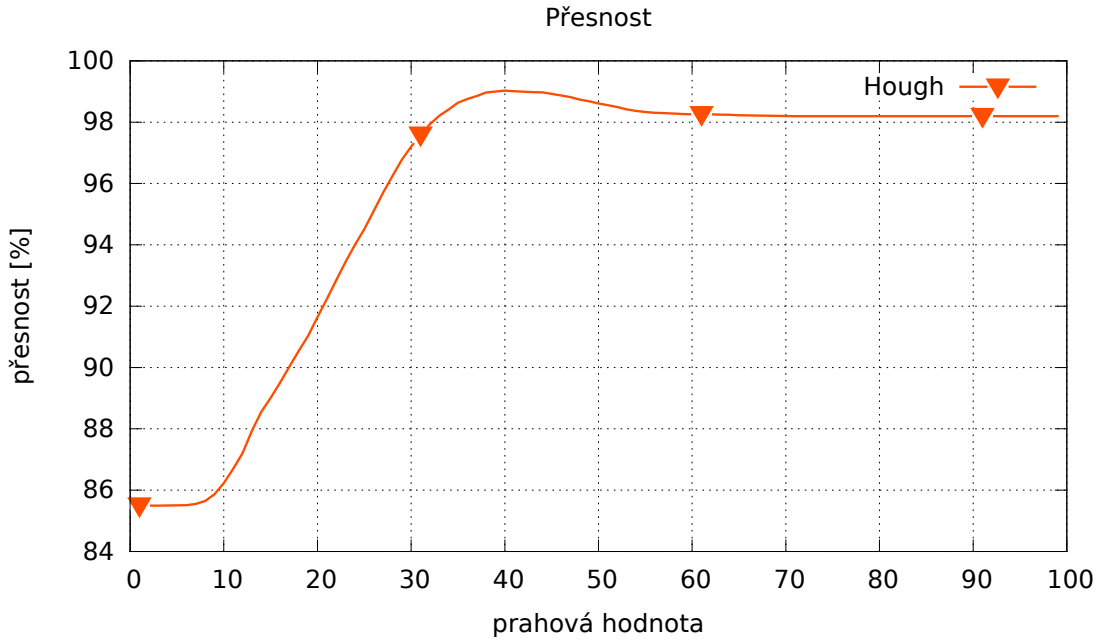
Dále byla vypočítána přesnost, které jsme dosáhli při použití jednotlivých prahových hodnot, a výsledná závislost byla vynesena do grafu zobrazeného v Obrázku 34. Z grafu je patrné, že ideální práh se pohybuje mezi hodnotami 35 a 45. Při měření získáme pro většinu prahových hodnot více jak 98 % přesnost. Navíc se výsledné hodnoty od sebe liší jen minimálně, což je způsobeno zahrnutím skutečně negativních detekcí TN do výpočtu. Hodnoty TN se oproti hodnotám ostatních veličin (TP, FN a FP) pohybují řádově v hodnotách deseti tisíců, což je demonstrováno v tabulce 4. Z tohoto důvodu se téměř všechny prahové hodnoty jeví jako vhodné a nejedná se tedy o ideální metriku pro zpřesňování detekce metod. Nadále již nebude veličina TN při testování uvažována.

Z výše uvedených důvodu byla pro zpřesňování detekce metod využita metrika F-measure. Výsledné hodnoty získané touto metrikou pro jednotlivé prahy jsou zobrazeny v grafu na Obrázku 35. Nejlepší hodnota je zvýrazněna v Tabulce 5.



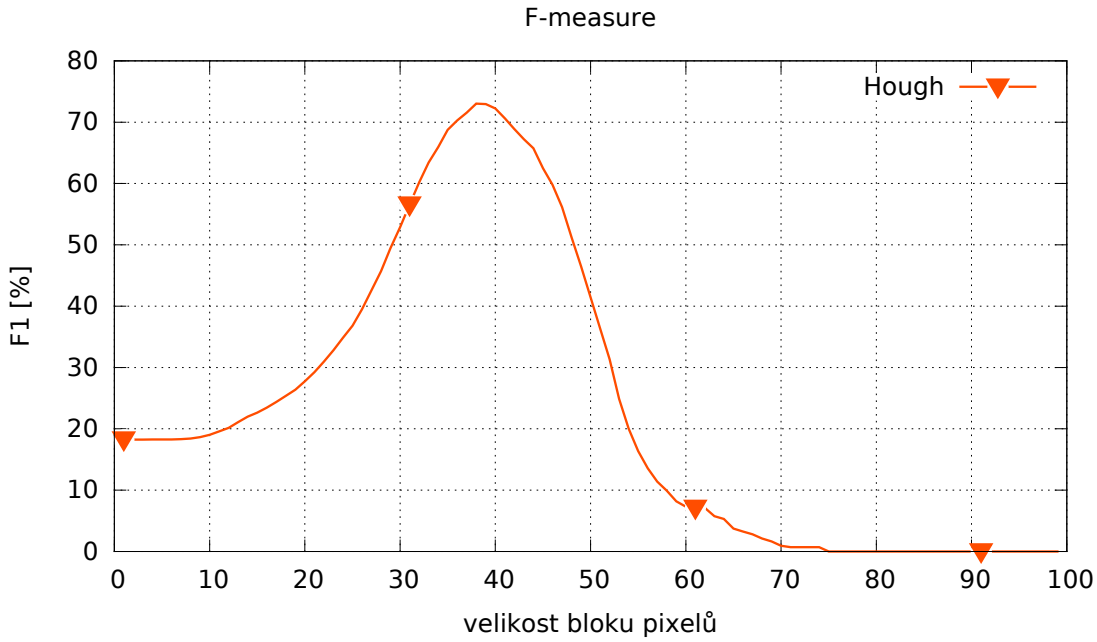
Obrázek 33: PR křivka zobrazující závislost precision a recall při hledání vhodného prahu pro poměr černých a bílých pixelů v testované oblasti, kde je očekávána plechovka. Vhodné hodnoty se v grafu vyskytují v pravém horním rohu.

Dále se pokusíme o zpřesnění detekce nastavením vhodné velikosti bloku pixelů, ze kterého bude počítána prahová hodnota pro právě zpracovávaný pixel. PR křivka znázorňující závislosti mezi precision a recall je vykreslena v Obrázku 36. Hodnoty naměřené pomocí metriky F-measure, které využijeme k určení vhodné velikosti bloku pixelů pro výpočet prahu, jsou zobrazeny v grafu na Obrázku 37. Nejlepší hodnota je zvýrazněna v Tabulce 6.



Obrázek 34: V grafu je zobrazena přesnost detekce jednotlivých prahových hodnot. Ideální hodnota prahu pro poměr černých a bílých pixelů v testované oblasti, kde je očekáván výskyt plechovky, se pohybuje mezi 35 až 45.

Po úpravě parametrů byla přesnost detekce zvýšena z 22.360 % na 74.437 %, což je demonstrováno zvýšením počtu skutečně pozitivních detekcí TP a snížením počtu falešně negativních FN.



Obrázek 35: Jednotlivé prahy pro poměr černých a bílých pixelů v oblasti byly ohodnoceny metrikou F-measure a vyneseny do grafu. Z grafu je patrné, že nejlepších výsledků bylo dosaženo mezi prahy 35 a 40.

Kaskáda slabých klasifikátorů s LBP příznaky

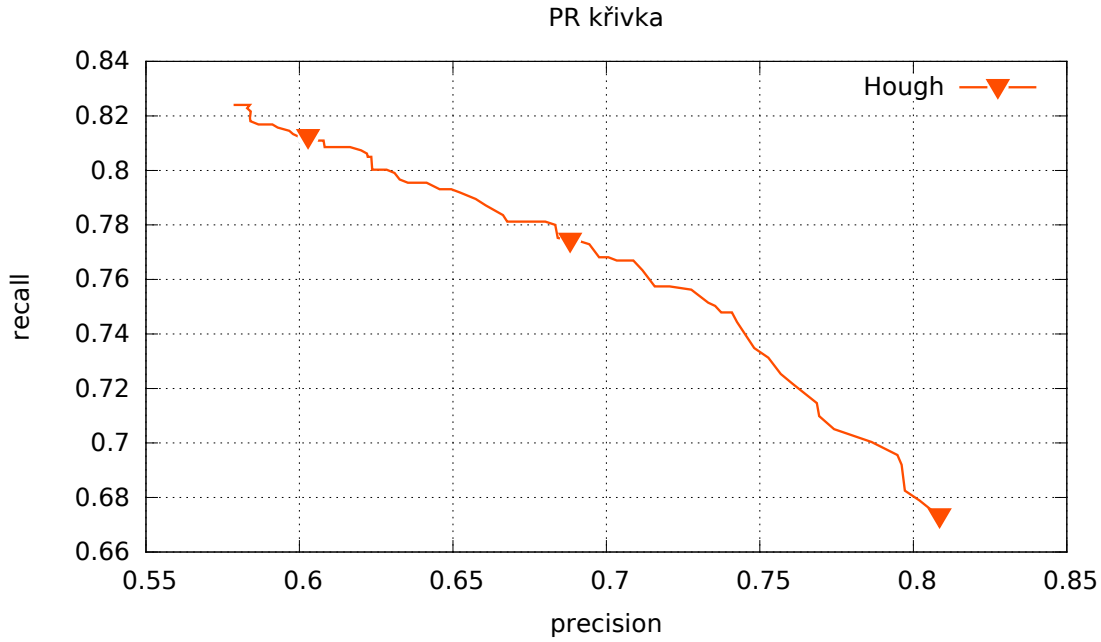
Stejně jako v minulé podsekci se pokusíme o zpřesnění detekce, tentokrát vybráním vhodné kaskády a nastavením vhodné hodnoty parametru `minNeighbors` v metodě `detectMultiScale()`. Tabulka 7 zobrazuje dosavadní úspěšnost metody, kterou bychom chtěli navýšit.

Dále se pokusíme o zpřesnění detekce vyhledáním vhodné hodnoty výše zmíněného parametru. Tento parametr ovlivňuje kvalitu detekovaných plechovek. Při zvýšení hodnoty tohoto parametru se bude na výstupu vyskytovat méně detekcí, ale s vyšší kvalitou. Tohle bude mít za následek menší počet falešně pozitivních detekcí. Pokud ovšem tato hodnota překročí určitý práh, začne ztrácet i skutečně pozitivní detekce. Ideální hodnoty prahu lze zjistit z Obrázku 39, kde je zobrazena PR křivka. Ideální hodnoty prahu se vyskytují v pravém horním rohu. Přesně tuto hodnotu určíme z grafu v Obrázku 40, kde byly jednotlivé hodnoty prahu ohodnoceny metrikou F-measure.

Po úpravě parametrů byla přesnost detekce zvýšena z 50.609 % na 65.155 %, což bylo zapříčiněno ohromným snížením falešně pozitivních detekcí FP. Snížen byl zároveň i počet TP detekcí, ale vzhledem k FP pouze nepatrně.

práh	TP	FN	FP	presicion	recall	F1 [%]
35	700	141	495	0.585	0.832	68.762
36	684	157	421	0.619	0.813	70.298
37	672	169	365	0.648	0.799	71.565
38	651	190	291	0.691	0.774	73.022
39	627	214	251	0.714	0.745	72.949
40	588	253	199	0.747	0.699	72.235

Tabulka 5: V tabulce je znázorněna úspěšnost detekce při použití vybraných prahových hodnot. Nejlepší hodnota prahu je 38.

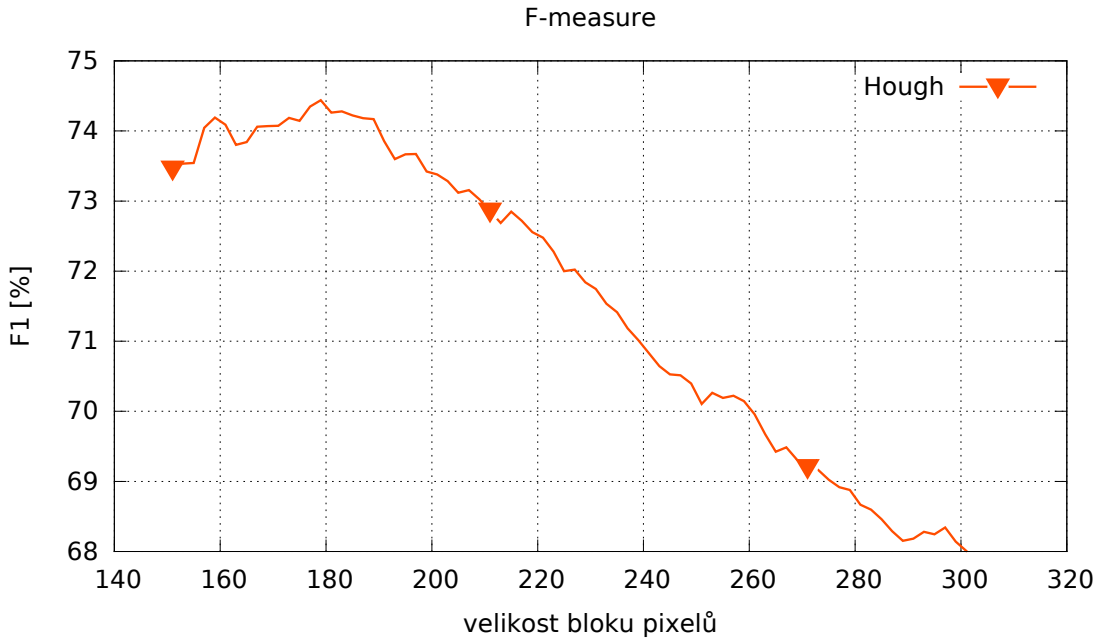


Obrázek 36: PR křivka zobrazující závislost precision a recall při hledání vhodné velikosti bloku pixelů, ze kterého bude počítán práh pro pixel. Ideální hodnoty prahu se vyskytují v pravém horním rohu.

7.4 Srovnání doby zpracování metod

V této sekci bude porovnávána doba zpracování metod na předpřipravených datasetech. Testy byly implementovány ve formě shellovských skriptů. Testování proběhlo na notebooku Lenovo T440s s procesorem Intel Core i7-4600U (4MB Cache, 2.10GHz).

Ze získaných výsledků, které jsou uvedeny v Obrázcích 41 - 45, je patrné, že metoda založená na Houghově transformaci má téměř konstantní dobu zpracování, která se pohybuje od $0.4s$ do $0.5s$ u všech datasetů. Doba zpracování je zde závislá na počtu čar hrací plochy, které byly zachyceny ve zpracovávaném snímku. S rostoucím



Obrázek 37: Jednotlivé velikosti bloků pixelů pro výpočet prahu byly ohodnoceny metrikou F-measure a vyneseny do grafu. Z grafu je patrné, že nejlepších výsledků bylo dosaženo mezi hodnotami 175 a 185. Přesné výsledky jdou uvedeny v tabulce 6.

počtem vodorovných i horizontálních čar hrací plochy tedy narůstá i počet průsečíků těchto čar, kde je následně testována přítomnost plechovky.

Metoda založená na strojovém učení má delší dobu zpracování. Tato doba zpracování se pohybuje okolo $0.6s$ u datasetů se zaostřenými a rozmazanými snímky, které jsou zobrazeny v Obrázcích 41 a 45. Větší rozpětí hodnot doby zpracování bylo zaznamenáno u datasetu, kde se vyskytuje pouze část plechovky. Výsledky jsou uvedeny v grafu na Obrázku 43. U datasetu, ve kterém se vyskytuje neznámý objekt, se doba zpracování pohybuje v největším rozpětí od $0.5s$ do $0.7s$. Výsledky jsou uvedeny v grafu na Obrázku 44.

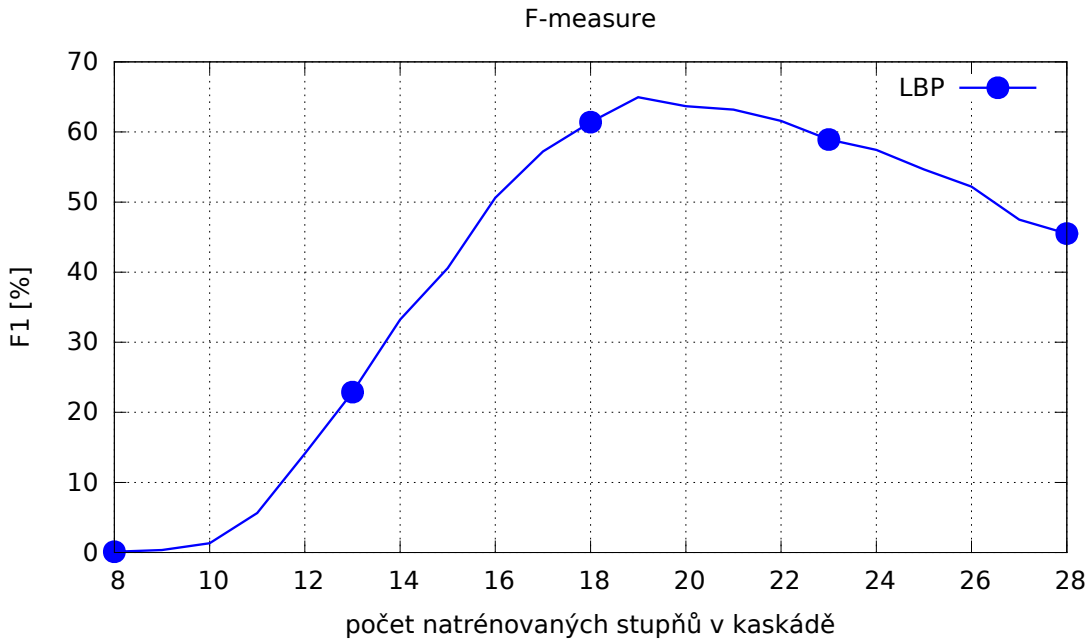
U datasetu se snímky, kde se vyskytuje pouze hrací plocha, bylo očekáváno nejrychlejší zpracování, jelikož při analýze snímku se zjišťuje, zdali právě zpracovávaný podobrázek vyhovuje všem klasifikátorům v kaskádě. Pokud při analýze podobrázek nevyhovuje nějakému klasifikátoru, je okamžitě zahozen a pokračuje se dalším podobrázkem, jak je nastíněno v Obrázku 13. Pravděpodobně zde bylo při analýze ověřováno mnoho klasifikátorů, než byl podobrázek zahozen, vlivem nízké kvality pořízených fotografií a mnoha různých intenzit světla v obraze. Důkazem tohoto tvrzení je fakt, že muselo být provedeno detailní trénování kaskády klasifikátorů, aby nedocházelo často k chybným detekcím plechovky v místech, kde se vyskytuje pouze část hrací plochy. Výsledky testů získané z datasetu, který obsahuje pouze

velikost bloku	TP	FN	FP	presicion	recall	F1 [%]
175	618	223	208	0.748	0.734	74.145
177	626	215	217	0.742	0.744	74.346
179	629	212	220	0.740	0.747	74.437
181	629	212	224	0.737	0.747	74.262
183	631	210	227	0.735	0.750	74.278
185	632	209	230	0.733	0.751	74.221

Tabulka 6: V tabulce je znázorněna úspěšnost detekce při použití vybraných velikostí bloků pixelů pro výpočet prahu pro pixel. Ideální velikost bloku pixelů je 179.

stupeň	práh	TP	FN	FP	precision	recall	F1 [%]
16	15	519	322	691	0.428	0.617	50.609

Tabulka 7: Tabulka znázorňuje původní přesnost metody (před vyhledáním vhodných parametrů). Atribut stupeň představuje počet natrénovaných stupňů v kaskádě. Atribut práh značí hodnotu parametru `minNeighbors`.

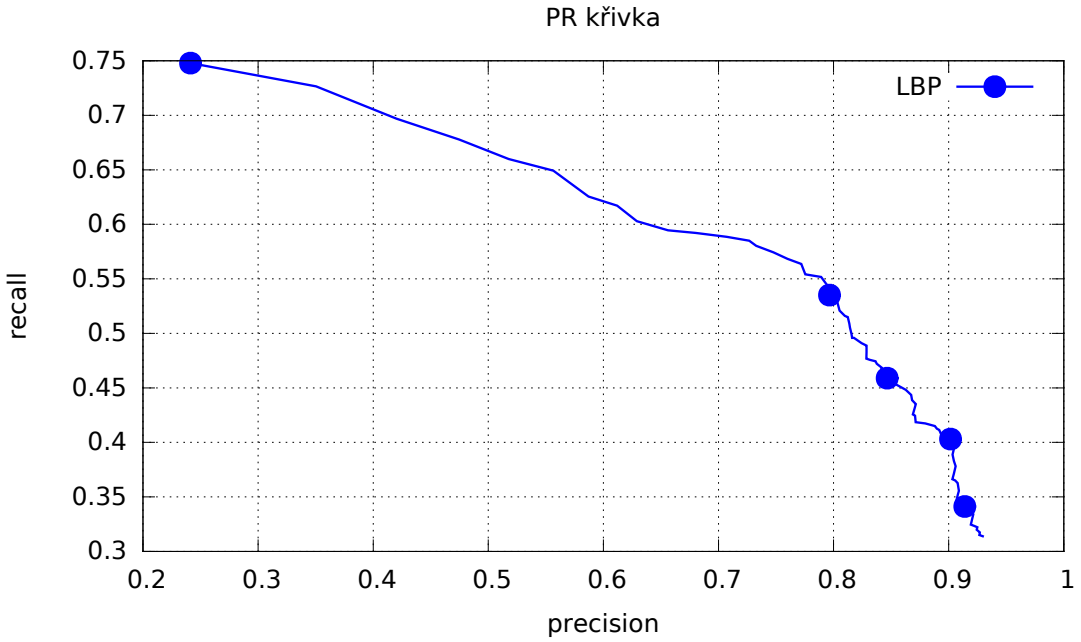


Obrázek 38: V grafu byly jednotlivé kaskády s určitým počtem natrénovaných stupňů ohodnoceny metrikou F-measure. Z grafu je zřejmé, že nejvyšší hodnoty nabývá kaskáda s natrénovanými 19 stupněmi.

hrací plochu, jsou zobrazeny na Obrázku 42.

stupeň	práh	TP	FN	FP	precision	recall	F1 [%]
19	15	483	358	163	0.747	0.574	64.963

Tabulka 8: Tabulka znázorňuje úspěšnost metody při použití kaskády s natrénovanými 19 stupněmi.

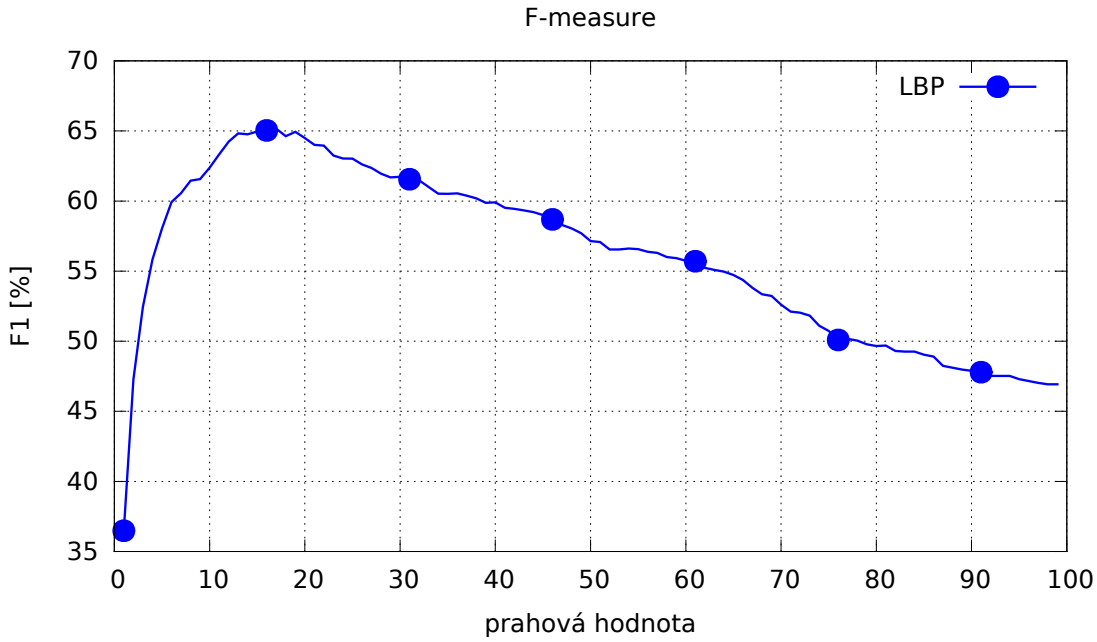


Obrázek 39: Křivka PR popisuje závislost precision a recall. Přijatelné hodnoty se v grafu vyskytují v pravém horním rohu

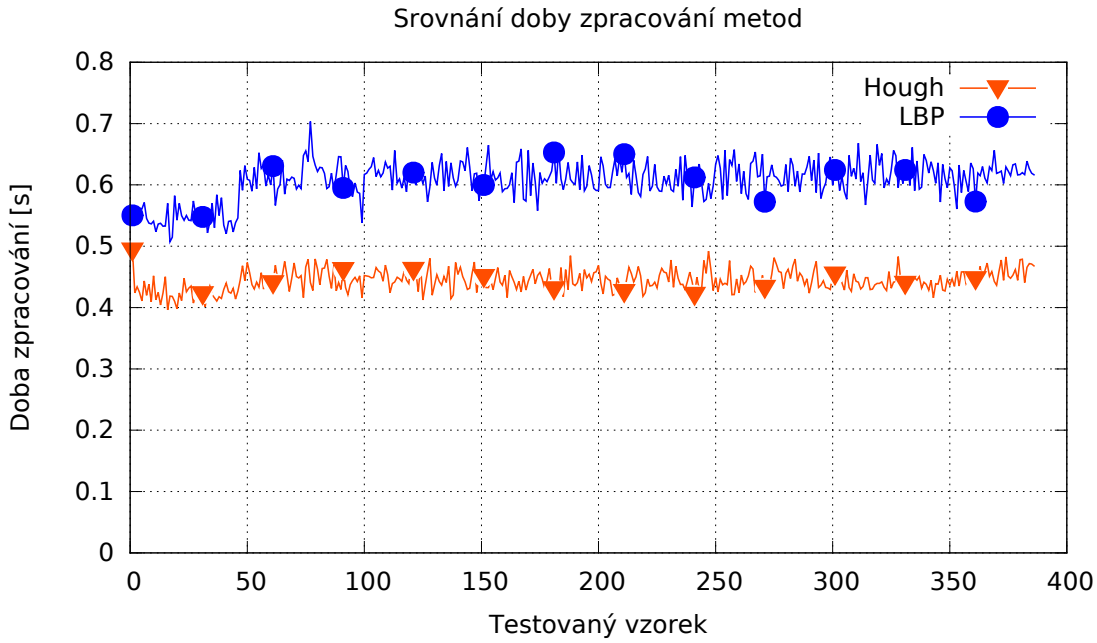
7.5 Srovnání detekce objektů

V této sekci bude provedeno srovnání úspěšnosti detekce objektů obou implementovaných metod. Při testování byly v metodách použity parametry, které byly v sekci Nastavení parametrů vyhodnoceny jako nejlepší.

Nejprve bylo provedeno testování úspěšnosti detekce na datasetu skládajícího se ze zaostřených snímků, kde byla zachycena kompletní plechovka. Zde se předpokládalo, že obě metody při detekci dosáhnou nejlepších výsledků, což bylo následně potvrzeno testováním. Znatelně lepších výsledků zde bylo dosaženo u metody založené na Houghově transformaci. Výsledky jsou uvedeny v Tabulce 10.



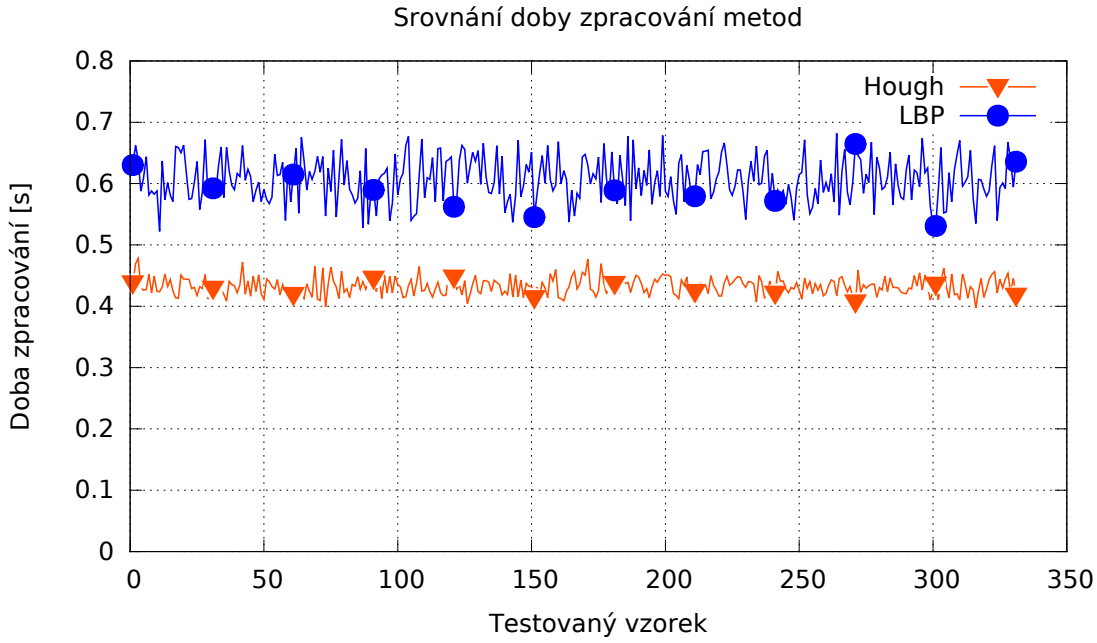
Obrázek 40: V grafu byly jednotlivé hodnoty prahu ohodnoceny metrikou F-measure. Z grafu je patrné, že nejlépe ohodnoceny byly prahy mezi hodnotami 10 a 20. Tyto hodnoty jsou uvedeny v Tabulce 9.



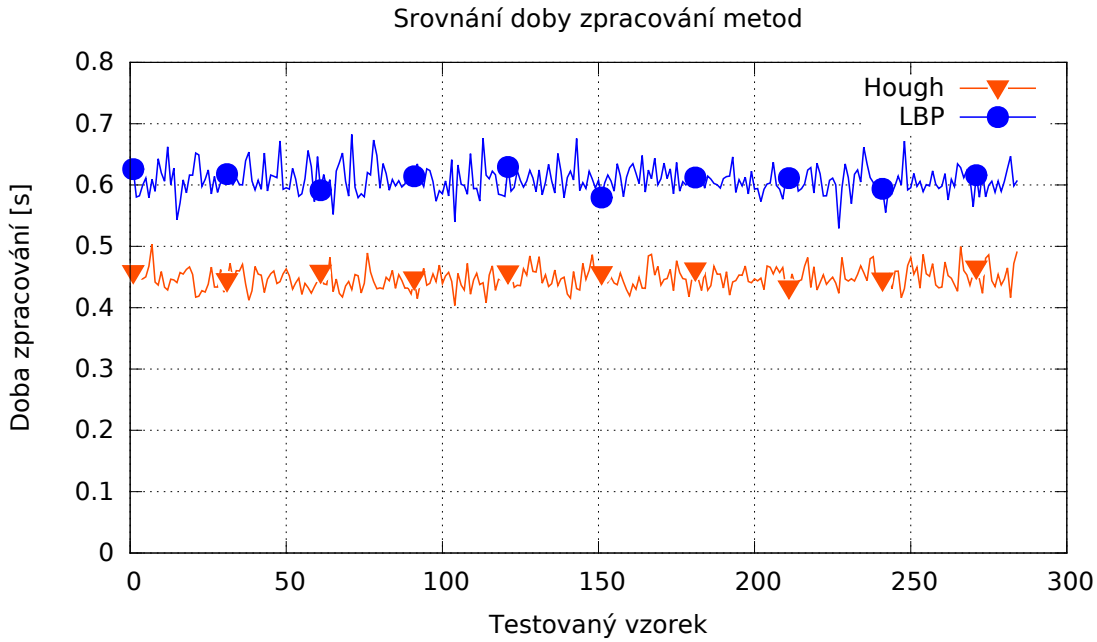
Obrázek 41: Srovnání doby zpracování metod na datasetu se zaostřenými fotkami.

práh	TP	FN	FP	precision	recall	F1 [%]
10	500	341	262	0.656	0.594	62.383
11	498	343	234	0.680	0.592	63.318
12	495	346	205	0.707	0.588	64.243
13	492	349	185	0.726	0.585	64.822
14	488	353	178	0.732	0.580	64.764
15	483	358	163	0.747	0.574	64.963
16	478	363	151	0.759	0.568	65.034
17	474	367	140	0.771	0.563	65.154
18	466	375	135	0.775	0.554	64.632
19	464	377	124	0.789	0.551	64.940
20	456	385	117	0.795	0.542	64.497

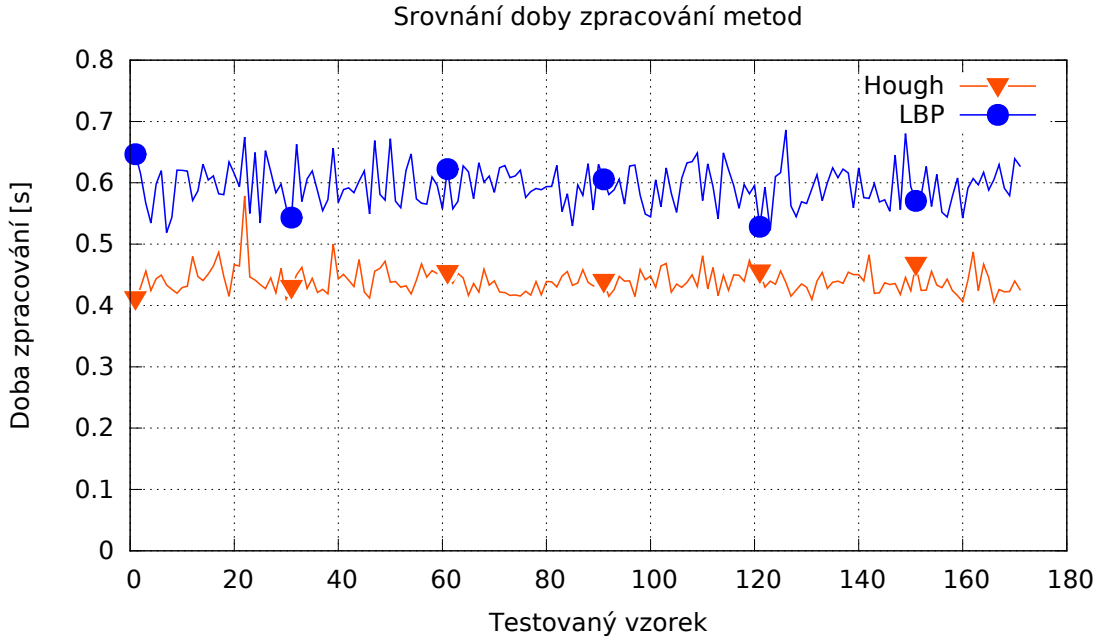
Tabulka 9: V tabulce je znázorněna úspěšnost detekce při použití vybraných prahových hodnot. Ideální hodnota prahu je 17.



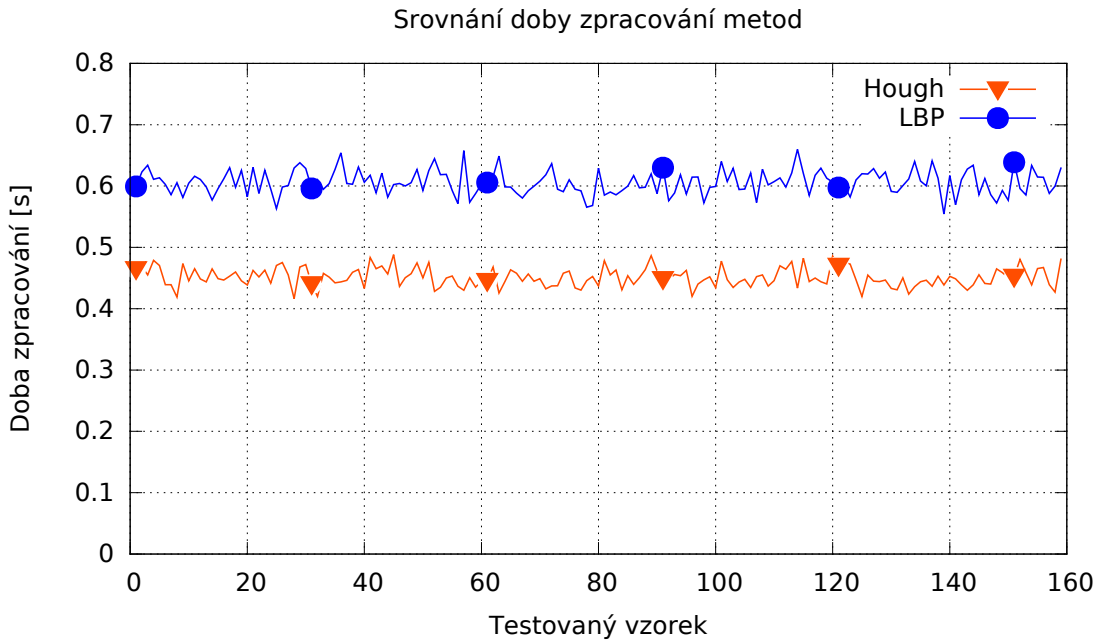
Obrázek 42: Srovnání doby zpracování metod na datasetu se snímky, na kterých se nevyskytuje žádný objekt.



Obrázek 43: Srovnání doby zpracování metod na datasetu se snímky, na kterých se vyskytuje pouze část objektu.



Obrázek 44: Srovnání doby zpracování metod na datasetu se snímky, na kterých se vyskytuje neznámý objekt (potenciální robot).



Obrázek 45: Srovnání doby zpracování metod na datasetu se snímky, které byly pořízeny ve vyšších rychlostech a v důsledku toho byly rozmazány.

metoda	**TP**	**FN**	**FP**	**F1** [%]
Hough	437	43	16	93.676
LBP	316	164	60	73.831

Tabulka 10: V tabulce je znázorněno srovnání úspěšnosti detekce implementovaných metod na datasetu sestávajícího ze zaostřených snímků. V datasetu se vyskytuje 380 snímků, ve kterých se dohromady nachází 480 plechovek. Úspěšnost byla poměřována na základě metriky F-measure. Lepších výsledků zde dosáhla metoda založená na Houghově transformaci.

Další testování bylo provedeno na datasetu zahrnujícího snímky, na kterých se nevyskytuje žádný objekt, tedy ze snímků s hrací plochou. U tohoto datasetu není možné ohodnotit úspěšnost pomocí metriky F-measure, protože se zde nevyskytuje žádný hledaný objekt a veličina TP musí zákonitě nabývat hodnoty nula. Z tohoto důvodu zde byl porovnáván počet falešně pozitivních detekcí FP. V tomto testu se ukázala být metoda založená na strojovém učení spolehlivější, jelikož chybovala pouze v 31 případech. Výsledky jsou uvedeny v Tabulce 11.

metoda	**TP**	**FN**	**FP**
Hough	0	0	62
LBP	0	0	31

Tabulka 11: V tabulce je znázorněno srovnání úspěšnosti detekce implementovaných metod na datasetu tvořeného ze snímků, na kterých se nevyskytuje žádný objekt. V datasetu se vyskytuje 332 snímků. Lepších výsledků zde dosáhla metoda založená na strojovém učení.

Jako třetí v pořadí byla otestována úspěšnost detekce na datasetu tvořeného ze snímků, kde byla zachycena pouze část plechovky. Metoda založená na Houghově transformaci podala velmi dobré výsledky, jelikož její úspěšnost závisí především na přesné detekci průsečíků hran, kde je následně testována přítomnost plechovky poměrem černých a bílých pixelů v oblasti. Poměr je víceméně zachováván ať se jedná o detekci kompletní plechovky, či pouze některé její části. Druhá metoda měla s detekcí výrazné problémy, což bylo samozřejmě způsobeno trénovací sadou dat. Trénovací sada zahrnovala různě natočené plechovky zabírané z mnoha různých úhlů. Pokaždé se však jednalo o záběr na kompletní plechovku. Řešením by bylo rozšířit trénovací sadu o nové vzorky skládající se z výřezů různě pootočených plechovek. Výsledky jsou uvedeny v Tabulce 12.

metoda	**TP**	**FN**	**FP**	**F1** [%]
Hough	327	179	96	70.398
LBP	139	367	65	39.154

Tabulka 12: V tabulce je znázorněno srovnání úspěšnosti detekce implementovaných metod na datasetu tvořeného ze snímků, kde je zachycena pouze část hledaného objektu. V datasetu se vyskytuje 284 snímků, ve kterých se dohromady nachází 506 plechovek. Úspěšnost byla poměřována na základě metriky F-measure. Lepších výsledků zde dosáhla metoda založená na Houghově transformaci.

Další testování proběhlo na datasetu skládajícího se ze snímků, kde se vyskytuje neznámý objekt. Jedná se o podobnou situaci jako v případě datasetu se snímky, na kterých se žádný objekt nevyskytoval. Opět zde bude poměřován pouze počet falešně pozitivních detekcí FP. Stejně jako v předchozím případě se v testu tohoto typu ukázala být spolehlivější metoda založená na strojovém učení, jelikož chybně detekovala pouze ve 34 případech. U druhé porovnávané metody byl neúspěch očekáván, jelikož nebyla implementována žádná dodatečná kontrola o detekovaném objektu. Jedinou informací o konkurenčním robotu je jeho maximální velikost, která nesmí přesáhnout určitou hranici. Tato informace je nedostatečná, pokud neznáme ani barevný odstín robota, který v extrémním případě může dokonce splývat s bílou plochou hrací plochy. Výsledky jsou uvedeny v Tabulce 13.

metoda	**TP**	**FN**	**FP**
Hough	0	0	135
LBP	0	0	34

Tabulka 13: V tabulce je znázorněno srovnání úspěšnosti detekce implementovaných metod na datasetu zahrnujícího snímky, na kterých se vyskytuje neznámý objekt (potenciální robot). V datasetu se vyskytuje 171 snímků. Lepších výsledků zde dosáhla metoda založená na strojovém učení.

Finální testování bylo provedeno na datasetu tvořeného ze snímků, které byly pořízeny během rychlejšího přesunu robota. Následkem toho byly tyto snímky rozmazány. Výsledky metod jsou velmi vyrovnané. O zhruba 2 % byly výsledky lepší u metody založené na Houghově transformaci, která detekovala více skutečně pozitivních detekcí, ale zároveň i více detekcí falešně pozitivních. Pokud však výsledky porovnáme s datasetem, kde se vyskytují zaostřené snímky, zjistíme, že rozmazání fotografie na detekci pomocí metody založené na strojovém učení, nemá téměř žádný vliv. Zatímco u druhé metody došlo k razantnímu zhoršení, což je způsobeno nepřesnější detekcí přímek, než tomu bylo u zaostřených snímků. Výsledky jsou uvedeny v Tabulce 14.

metoda	**TP**	**FN**	**FP**	**F1** [%]
Hough	176	72	73	70.824
LBP	143	105	25	68.75

Tabulka 14: V tabulce je znázorněno srovnání úspěšnosti detekce implementovaných metod na datasetu skládajícího se z rozmazaných snímků. V datasetu se vyskytuje 160 snímků, ve kterých se dohromady nachází 248 plechovek. Úspěšnost byla poměřována na základě metriky F-measure. Lepších výsledků zde dosáhla metoda založená na Houghově transformaci.

8 Závěr

Cílem diplomové práce bylo vyvinout počítačové vidění pro robota v podobě detekce objektů na hrací ploše a to takovým způsobem, aby detekce proběhla v co možná nejkratším čase. Za tímto účelem jsem navrhl a následně naimplementoval dvě metody. První metoda je založena na Houghově transformaci a druhá na natrénované kaskádě slabých klasifikátorů, kde lze zvolit Haarovy nebo LBP příznaky. Metody jsem volil s ohledem na kritéria rychlosti zpracování a přesnosti detekce objektů.

Hlavní náplní mé práce bylo zhodnocení implementovaných metod a následný výběr metody, která bude použita do provozu. Před samotným srovnáním metod jsem pomocí metriky F-measure vyhledal vhodné parametry za účelem zvýšení úspěšnosti detekce. Při hledání ideálního parametru bylo nutné otestovat úspěšnost metody se všemi platnými hodnotami parametru. Doba testování jednoho parametru se pohybovala okolo **14 hodin**. Při hledání vhodných parametrů bylo zjištěno, že kaskáda slabých klasifikátorů s LBP příznaky má ve všech ohledech lepší výsledky detekce objektů než při použití Haarových příznaků. Z tohoto důvodu byla ke srovnání s metodou založenou na Houghově transformaci vybrána pouze kaskáda slabých klasifikátorů s LBP příznaky.

Následně jsem otestoval dobu zpracování metod, přičemž jsem zjistil, že obě metody jsou schopné zpracovat snímek v rozumném čase vzhledem k velikosti snímku. U první metody se doba zpracování pohybuje mezi **0.4 a 0.5 sekundami** a to pro jakýkoli typ snímku. U druhé metody je doba zpracování delší, pohybuje se od **0.5 do 0.7 sekund**, kdy tato doba je závislá na typu snímku. Nejdéle jsou zpracovávány snímky, kde se vyskytuje neznámý objekt (potenciální robot).

Další srovnání proběhlo na základě úspěšnosti detekce objektů v obraze, které bylo rovněž ohodnoceno metrikou F-measure. Kritériem úspěšnosti je co možná nejvyšší počet skutečně pozitivních detekcí v kombinaci s nízkým počtem falešných detekcí.

První metoda (Houghova transformace) dosáhla lepších výsledků při detekci objektů na zaostřených a rozmazaných snímcích. Stejně tomu bylo i v případě snímků, kde byla zachycena pouze část hledaného objektu. Naopak horších výsledků dosáhla na snímcích, kde bylo testováno rozpoznání nepřítomnosti hledaného objektu. Obzvláště tomu tak bylo v případech, kdy se v obraze vyskytoval neznámý objekt, který byl v rámci detekce považován za hledaný objekt.

Na rozdíl od první metody nemá druhá metoda (LBP) problém s chybnou detekcí cizího objektu. Jejím nedostatkem je však nižší počet správně detekovaných skutečných objektů v obraze. To je způsobeno příliš rozdílnými světelnými podmínkami ve snímcích mezi trénovací a testovací sadou.

Přestože se jedná o problém, který lze velmi snadno vyřešit rozšířením trénovací sady o další snímky pořízené v co možná nejvíce různých světelných podmínkách, navrhuji do provozu nasadit na robota **první metodu (Houghova transformace)**. Tato metoda je rychlejší a je zároveň více flexibilní díky předzpracování obrazu pomocí adaptivního prahování. Výsledkem je vyšší odolnost na změny prostředí

a světelných podmínek okolí. Zároveň je úspěšná detekce plechovky podmíněna pouze jejími rozměry a ne zahrnutím všech možných typů plechovky do trénovací sady jako v případě druhé metody.

Pokud by nakonec byla zvolena pro nasazení na robota druhá metoda (LBP), je možně využít sadu pomocných nástrojů, které jsem implementoval při vytváření trénovací a testovací sady vzorků.

Na závěr jsem se pokusil odhadnout pozici vyhledaného objektu v reálném prostoru. Jelikož jsem měl k dispozici pouze snímek z jediné kamery, jedná se jen o hrubý odhad vzdálenosti objektu od kamery a úhlu, o který je robot vzhledem k objektu pootočen, který byl získán na základě polárních souřadnic. Testování přesnosti určování pozice prokázalo, že se skutečně nejedná o proveditelný úkol. Vzdálenost objektu od kamery byla odhadnuta s relativní odchylkou 6.23 %, přičemž se v několika případech určená vzdálenost od skutečné vzdálenosti lišila až o $\pm 5\,cm$. Velikost úhlu, o který byl robot vzhledem k plechovce pootočen, byla odhadnuta s průměrnou relativní odchylkou 13.981 %, přičemž se v několika případech odhadnutá hodnota úhlu lišila od skutečné hodnoty až o $\pm 3\,^\circ$. Řešením tohoto problému může být nasazení ještě minimálně jedné kamery na robota.

Možným navazujícím výzkumem by mohlo být využití dalších metod strojového učení k detekci objektů na hrací ploše. Zajímavé výsledky bychom mohli získat při natrénování kaskády slabých klasifikátorů s využitím Haar i LBP příznaků. Dále by bylo možné provést zpřesnění detekce objektů (neznámého i hledaného) použitím snímků z více kamer a určení jejich přesné polohy v reálném prostoru.

9 Reference

[Akram a Ismail, 2013] AKRAM, A. a Ismail, A. *Comparison of Edge Detectors.*, 2013.

[Bradski, 2000] BRADSKI, G. *The OpenCV Library.* Dr. Dobb's Journal of Software Tools, 2000.

[Bradski a Kaehler, 2008] BRADSKI, G. a KAEHLER, A. *Learning OpenCV.* O'Reilly Media, Inc., 2008. ISBN 978-0-596-51613-0.

[Canny, 2008] CANNY, J. *A computational approach to edge detection.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986.

[scikit-image, 2016] scikit-image *Adaptive Thresholding* [online]. 2016 [cit. 2.5.2016]. Dostupné z `http://scikit-image.org/docs/dev/auto_examples/segmentation/plot_threshold_adaptive.html`.

[scikit-image, 2016] scikit-image *Local Binary Pattern for texture classification* [online]. 2016 [cit. 14.5.2016]. Dostupné z `http://scikit-image.org/docs/dev/auto_examples/plot_local_binary_pattern.html`.

[Doman, Deguchi, Takahashi, Mekada, Ide a Murase, 2009] DOMAN, K., DEGUCHI, D., TAKAHASHI, T., Mekada, Y., Ide I. a Murase, H. *Construction of Cascaded Traffic Sign Detector Using Generative Learning.*, 2009.

[Duda a Hart, 1972] DUDA, R., O. a HART, P., E. *Use of the Hough Transformation to Detect Lines and Curves in Pictures.* ACM, 1972.

[Fisher, Perkins, Walker a Wolfart, 2003] FISHER, R., PERKINS, S., WALKER, A. a WOLFART, E. *Gaussian Smoothing* [online]. 2003 [cit. 2.5.2016]. Dostupné z `http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm`.

[Green, 2002] GREEN, B. *Canny Edge Detection Tutorial.* 2002.

[Han, Han a Hahn, 2009] HAN, S., HAN, Y., Hahn, H. *Vehicle Detection Method using Haar-like Feature on Real Time System.*, 2009.

[Hozman, 2003] HOZMAN, J. *Základní metody předzpracování obrazu.* katedra radioelektroniky FEL ČVUT v Praze, 2003.

[Jiang, Hornegger, Koch, 2014] JIANG, X., HORNEGGER, J., KOCH, R. *Pattern Recognition.*, 2014. ISBN 978-3-319-11752-2.

[Manning, Raghavan a Schütze, 2008] MANNING, Christopher, D., RAGHAVAN, P. a SCHÜTZE, H. *Introduction to Information Retrieval.* Cambridge University Press, 2008 ISBN 9780521865715.

[Moeslund, 2009] MOESLUND, T. *Canny Edge Detection.* 2009.

[NumPy, 2005] NumPy. *NumPy* [online]. 2005 [cit. 2.5.2016]. Dostupné z http://www.numpy.org/.

[Perďoch, 2008] PERĎOCH, M. *Hledání parametrického popisu objektů pomocí Houghovy transformace* [online]. 2008 [cit. 2.5.2016]. Dostupné z http://cmp.felk.cvut.cz/cmp/courses/ZS1/Cviceni/cv5/hough.htm.

[Pietikäinen, 2010] PIETIKÄINEN, M. *Local Binary Patterns* [online]. 2010 [cit. 7.5.2016]. Dostupné z http://www.scholarpedia.org/article/Local_Binary_Patterns.

[Pietikäinen, Hadid, Zhao a Ahonen, 2011] PIETIKÄINEN, M., HADID, A., ZHAO, G. a AHONEN, T. *Computer Vision Using Local Binary Patterns.* Springer London, 2011. ISBN 978-0-85729-748-8.

[PixelsTech, 2012] PixelsTech. *Gaussian Blur Algorithm* [online]. 2012 [cit. 2.5.2016]. Dostupné z http://www.pixelstech.net/article/1353768112-Gaussian-Blur-Algorithm.

[Raspberry Pi, 2016] Raspberry Pi. *Camera Module* [online]. 2016 [cit. 9.5.2016]. Dostupné z https://www.raspberrypi.org/documentation/hardware/camera/README.md.

[Rosebrock, 2015] ROSEBROCK, A. *Local Binary Patterns with Python & OpenCV* [online]. 2010 [cit. 7.5.2016]. Dostupné z http://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/.

[Šafařík, 2011] ŠAFAŘÍK, P. *Houghova transformace pro detekci přímek na obrázku* [online]. 2011 [cit. 2.5.2016]. Dostupné z http://petos.cz/2011/houghova-transformace-pro-detekci-primek-na-obrazku.

[Turkel, 2011] TURKEL, E. *Thresholding* [online]. 2011 [cit. 2.5.2016]. Dostupné z http://www.math.tau.ac.il/ťurkel/notes/threshold.pdf.

[Tým AiStorm, 2016] Tým AiStorm *Mobilní robot K4* [online]. 2016 [cit. 14.5.2016]. Dostupné z https://aistorm.mendelu.cz/cz/projekty/k4.

[Viola a Jones, 2001] VIOLA, P. a JONES, M. *Rapid Object Detection using a Boosted Cascade of Simple Features.* Computer Vision and Pattern Recognition, 2001.

[Vlach, 2011] VLACH, J. *Hledání úseček a kružnic s využitím Houghovy transformace při zpracování obrazu v LabView.* 2011.

[Vránová, Horák, Krátká, Hendrichová a Kovaříková, 2009] VRÁNOVÁ, J., HORÁK, J., KRÁTKÁ, K., HENDRICHOVÁ, M. a KOVAŘÍKOVÁ, K. *ROC analýza a využití analýzy nákladů a přínosů k určení optimálního dělícího bodu.* 2009.

[WaveMetrics, 2015] WaveMetrics *Image Threshold* [online]. 2015 [cit. 2.5.2016]. Dostupné z `https://www.wavemetrics.com/products/igorpro/imageprocessing/thresholding.htm` .

[Wikipedia, 2016] Wikipedia *Thresholding (image processing)* [online]. 2016 [cit. 2.5.2016]. Dostupné z `https://en.wikipedia.org/wiki/Thresholding_%28image_processing%29` .

[Wilson a Fernandez, 2006] WILSON, P., I., FERNANDEZ, J. *Facial Feature Detection Using Haar Classifiers.* J. Comput. Sci. Coll., 2006.