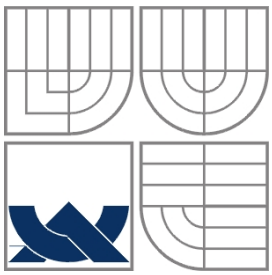


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

METODY ROZPOZNÁVÁNÍ OBJEKTŮ SNÍMANÝCH KAMEROU PRO ÚČELY ROBOTICKÉ MANIPULACE

METHODS OF OBJECT RECOGNITION FOR PURPOSES OF THE ROBOTIC MANIPULATION

BAKALÁŘSKÁ PRÁCE
BACHELOR THESIS

AUTOR PRÁCE **FILIP DALECKÝ**
AUTHOR

VEDOUCÍ PRÁCE **ING. JIŘÍ KOVÁŘ**
SUPERVISOR

BRNO 2013

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2012/2013

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Filip Dalecký

který/která studuje v **bakalářském studijním programu**

obor: **Strojní inženýrství (2301R016)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Metody rozpoznávání objektů snímaných kamerou pro účely robotické manipulace

v anglickém jazyce:

Methods of object recognition for purposes of the robotic manipulation

Stručná charakteristika problematiky úkolu:

Robotická manipulace objektů, které nejsou prostorově uspořádány v předem daném vzoru, je závislá na metodách rozpoznávání obrazu. Tato práce je rozdělena na dvě části. První část práce bude tvořena rešeršní studii tématu. V druhé části bude jedna vybraná metoda realizována v některém z programovacích jazyků.

Cíle bakalářské práce:

Prostudujte danou problematiku a nejpoužívanější metody popiště v rešeršní části práce. Vyberte jednu metodu a tu realizujte.

Seznam odborné literatury:

[1] Theodoridis, S., Pikrakis, A., Koutroumbas, K., Cavouras, D.: Introduction to Pattern Recognition: A Matlab Approach, 2010, Academic Press
[2] Theodoridis, S., Koutroumbas, K.: Pattern recognizion, 2006, Elsevier
[3] Hanák, J, "C# 3.0 - Programování na platformě .NET 3.5", Zoner Press, 2009, ISBN: 978-80-7413-046-5
[4] Raghavendra, R., Dorizzi, B., Rao, A., Kumar, G.H., "Particle swarm optimization based fusion of near infra red and visible images for improved face verification", vol. 44, issue 2, February 2011, pp. 401-411

Vedoucí bakalářské práce: Ing. Jiří Kovář

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2012/2013.

V Brně, dne 11.10.2012

L.S.

Ing. Jan Roupec, Ph.D.	prof. RNDr. Miroslav Doupovec, CSc., dr. h. c.
Ředitel ústavu	Děkan fakulty

ABSTRAKT

Tato bakalářská práce se zabývá problémem detekce nahodile umístěných neorientovaných objektů pro účely robotické manipulace. Hlavním cílem první části této práce je, seznámit se se základními metodami zpracování obrazu. V následující části je vybrána vhodná metoda detekce obdélníkových objektů, která je realizována v prostředí NI LabVIEW. Dále je popsána konstrukce, kinematika a řízení robotického manipulátoru, který byl součástí experimentu. V poslední kapitole jsou rozebrány a vyhodnoceny výsledky experimentu.

ABSTRACT

This bachelor thesis deals with the detection issue of randomly placed unoriented objects for purposes of robotic manipulation. The first part of this paper is aimed to get acquainted with the basic methods of image processing. In the next section, a choice of the suitable method is made to create software for the rectangular shaped objects detection in NI LabVIEW. Subsequently, the robotic arm manipulator construction and related issues, such as kinematics, servo controlling, are briefly described. In the reminder of this paper, some experimental results are shown and evaluated.

KLÍČOVÁ SLOVA

Detekce objektů, zpracování obrazu, robotická manipulace, sériový manipulátor, NI LabVIEW.

KEYWORDS

Object detection, image processing, robotic manipulation, serial manipulator, NI LabVIEW.

PROHLÁŠENÍ O ORIGINALITĚ

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně dle pokynů vedoucího a s použitím uvedené odborné literatury.

V Brně, dne 9. 5. 2013

BIBLIOGRAFICKÁ CITACE

DALECKÝ, F. *Metody rozpoznávání objektů snímaných kamerou pro účely robotické manipulace*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2013. 41 s. Vedoucí bakalářské práce Ing. Jiří Kovář.

PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu práce Ing. Jiřímu Kovářovi za odborné vedení, jeho ochotu, čas a trpělivost při vytváření této bakalářské práce. Veliké dík patří také rodině a přítelkyni, jež mi byly v dobách psaní oporou.

Obsah:

1 ÚVOD

Současným trendem v moderním průmyslu je zvyšovat přesnost, rychlost a efektivitu výroby. Lidská práce bývá často nahrazována automatizovanými výrobními linkami a to nejen z důvodu snížení nákladů a urychlení výroby, ale především z toho důvodu, že na spoustu úkonů už samotná lidská síla nestačí nebo zde existuje vysoké riziko úrazu.

Jedním z úkolů robota je přemísťování dílů či výrobků z jednoho místa na druhé. U dílů putujících na dopravníku nebo uložených v zásobníku nelze ve všech případech zajistit přesně orientované řazení. Tam, kde to není možné, je nutné pracovní prostor snímat a pomocí zpracování obrazu získat informaci o jejich poloze a natočení, která je dále předána robotickému manipulátoru.

Tato práce se v teoretické části zaměřuje na popis vybraných přístupů ke zpracování obrazu, ze kterých bude vybrána vhodná metoda na uskutečnění experimentu. Experiment spočívá ve třídění barevných kostek, které budou nahodile pohozené v pracovním prostoru. Pracovní prostor bude snímaný webkamerou, obraz z webkamery bude následně zpracován v softwaru NI LabVIEW. Součástí této práce bude také výroba jednoduchého modelu manipulátoru s pěti stupni volnosti, který zajistí přemístění barevných objektů do příslušných kontejnerů. Výpočet kinematiky a řízení manipulátoru budou rovněž zpracovány pomocí NI LabVIEW.

Výzkumné a vývojové centra nyní v této disciplíně řeší mnohem náročnější problém a tím je manipulace neorientovaných nahodile rozprostřených dílů z kontejneru. Tento proces je nazýván anglickým souslovím bin picking. V tomto případě přichází v úvahu použití 3D technologie, která dokáže získat mnohem kvalitnější informaci o poloze dílců v krabici, než je tomu u konvenčních metod. Celý proces je velmi složitý a bude předmětem vývoje ještě po několik dalších let.

2 MOTIVACE K PRÁCI

Hlavním cílem této práce je seznámení se s problematikou zpracování obrazu a robotické manipulace objektů. Toto téma jsem si vybral především díky svému zájmu a také příležitosti rozšířit si znalosti v této problematice.

Snímání pracovní plochy je velmi důležité pro samotnou práci manipulátoru, bez něj by musely být veškeré pozice a natočení objektů zadány uživatelem nebo jiným způsobem. Existuje velké množství metod na detekci objektů v obraze, nicméně žádná není natolik univerzální, aby ji bylo možné aplikovat pro libovolnou situaci. Díky tomu je zde prostor pro inovace a nová řešení. Praktická část práce se zaměřuje na detekci obdélníkových objektů a to zjištění jejich polohy, natočení a také barvy. Experiment poté prověří spolehlivost detekce na jednoduchém modelu manipulátoru, který bude třídit a přesouvat barevné kostky z polystyrenu.

Reálné využití je ovšem o mnoho širší než jen samotná manipulace modelových kostek. Tato metoda zpracování obrazu by se mohla například používat ke kontrole osazení obdélníkových (po snadné úpravě i kruhových) součástek tištěných spojů.

3 ZPRACOVÁNÍ OBRAZU

Celý proces zpracování a rozpoznávání obrazu se skládá z několika základních kroků. Rozdělení jednotlivých kroků se může lišit dle dané aplikace, a proto je možné narazit v různých publikacích na jiné dělení.

Snímání a digitalizace obrazu – získaný obraz, např. z kamery, skeneru, lékařských zařízení, je v číselné podobě uložen do počítače. Vstupní informací může být jas, nebo v případě barevného snímání, několik spektrálních složek, ty se liší od použitého barevného modelu (RGB, HSL, YUV, CMYK apod.). Digitalizace je pojem používaný pro převod analogového signálu do diskrétního tvaru. Digitální obraz je ekvivalentem spojité funkce *f(x,y),* kde *x* a *y* jsou souřadnice v obraze a funkční hodnota odpovídá např. jasu. Je získán vzorkováním obrazu do matice *MxN* bodů a kvantováním do *K* úrovní. Jeden prvek matice je nazýván pixel.

Předzpracování obrazu – nasnímaný obraz může trpět mnoha neduhy, je to především šum, nevhodné světelné podmínky nebo zkreslení obrazu zapříčiněné optikou, nebo špatným úhlem snímání kamery. Cílem předzpracování obrazu je tedy eliminovat šum, perspektivu a potlačit vliv osvětlení.

Segmentace obrazu – v této fázi zpracování obrazu je úkolem rozlišit námi hledané objekty od pozadí, je nejsložitější z celého procesu a bude blíže popsána v následujících podkapitolách včetně popisu některých používaných metod.

Popis objektů – dalším krokem je popis obrazu nebo také popis objektů. Objekty mohou být popsány kvantitativně a kvalitativně. Do kvantitativního popisu se řadí například velikost objektů, obvod, natočení, barva atd. V kvalitativním přístupu jsou popisovány relace mezi objekty a jejich tvarové vlastnosti.

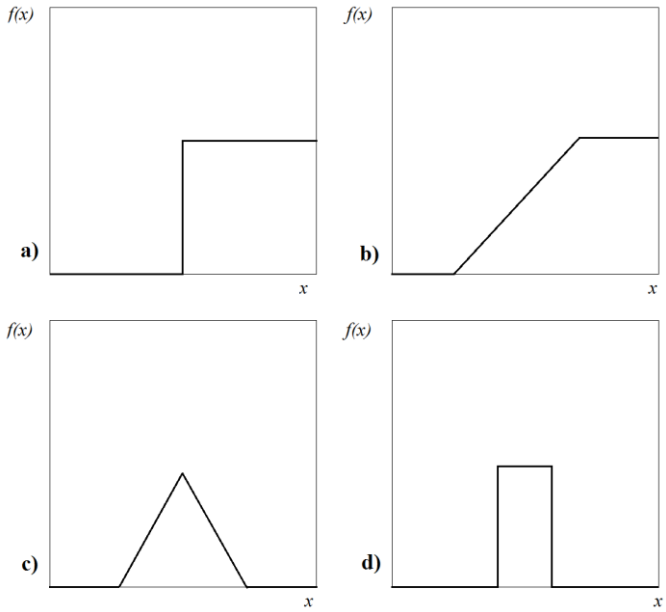
Klasifikace objektů – ve většině případů se jedná o zařazení objektů nalezených v obraze do skupiny tříd na základě vlastností získaných popisem objektů. Klasifikace může být uskutečněna přímo porovnáním reálných hodnot s hodnotami, které má nasnímaný objekt splňovat, jako jsou např. rozměry na snadno měřitelných místech (rohy, hrany, díry), barva objektu atd., anebo metodou porovnávání s referenčním modelem [1][3].

3.1 Segmentace

Jedná se o proces, kdy dochází k rozdělení obrazu na jednotlivé části podle charakteristických vlastností, jako je intenzita bodu, barva nebo textura. Cílem je, aby výsledné segmenty korespondovaly s hledanými objekty v obraze. V reálných situacích není možné dosáhnout úplně přesné segmentace, ale je snahou co nejvíce se jí přiblížit. Po segmentaci často vznikají segmenty, které jsou buď příliš malé, nebo nekorespondují s objekty. Proto je dále nutné některé segmenty spojit, případně rozdělit. Dalším krokem může být tzv. labeling, tento proces má za úkol přiřadit jednotlivým objektům vlastní index, to značně usnadňuje následující práci s objekty [2][3].

3.2 Metody založené na detekci hran

Významným jevem vyskytujícím se v obraze jsou prudké lokální změny intenzity jasu nebo barvy, právě takovými jako jsou hrany. Hrany jsou velmi důležité pro vizuální vnímání a interpretaci obrazů. Důležitost hran potvrzuje fakt, že pro lidské vnímání stačí nákres tvořený pouze několika čarami k tomu, aby byl jednoznačně popsán objekt nebo scéna. Existuje spousta metod na hledání hran v obraze, i přes to je detekce hran v reálném prostředí dodnes značným problémem. Snímaná scéna je obvykle zašuměná nebo nemá vhodné světelné podmínky. Ideální hrana má skokový průběh, nicméně v reálných scénách často bývá průběh postupný. Některé typy hran jsou znázorněny v následujícím obrázku (o*br. 1*) [2].



Obr. 1 Typy hran: a) step; b) ramp; c) roof; d) line.

3.2.1 Gradient-based detekce hran

Gradientní metody detekce hran vycházejí z principu, že v místech, kde se nachází hrana, dochází k největší změně velikosti gradientu. Nejdříve je proveden výpočet parciální derivace obrazu podle osy x a osy y, z nich je vypočítána velikost a orientace vektoru [2].

$$\frac{\partial G}{\partial x} \text{ a } \frac{\partial G}{\partial y} \tag{3.1}$$

jsou parciální derivace obrazové funkce podle osy x a osy y. Funkce

$$\nabla G(x,y) = \left(\frac{\partial G}{\partial x}, \frac{\partial G}{\partial y} \right) \tag{3.2}$$

se nazývá gradient obrazové funkce *G(x, y)*, jehož složky tvoří jednotlivé parciální derivace. Velikost gradientu a orientace je pak dána ze vztahů

$$\left| \nabla G(x,y) \right| = \sqrt{\left(\frac{\partial G}{\partial x} \right)^2 + \left(\frac{\partial G}{\partial y} \right)^2} \tag{3.3}$$

$$\theta = \arctan\left(\frac{\frac{\partial G}{\partial y}}{\frac{\partial G}{\partial x}} \right) \tag{3.4}$$

Pro zjištění gradientu lze k výpočtu přistupovat jako ke konvolučnímu filtrování signálu. Konvoluční jádro tvoří jeden z hranových operátorů. Hranové operátory sloužící ke zjištění aproximovaných hodnot první derivace používají dvě matice o rozměru 3x3 – jednu pro derivaci podle osy x a druhou pro derivaci podle osy y. Nejsou neobvyklé ani operátory využívající matice vyšších řádů. Mezi významné hranové operátory patří Robertsův, Sobelův a Prewittův operátor [3].

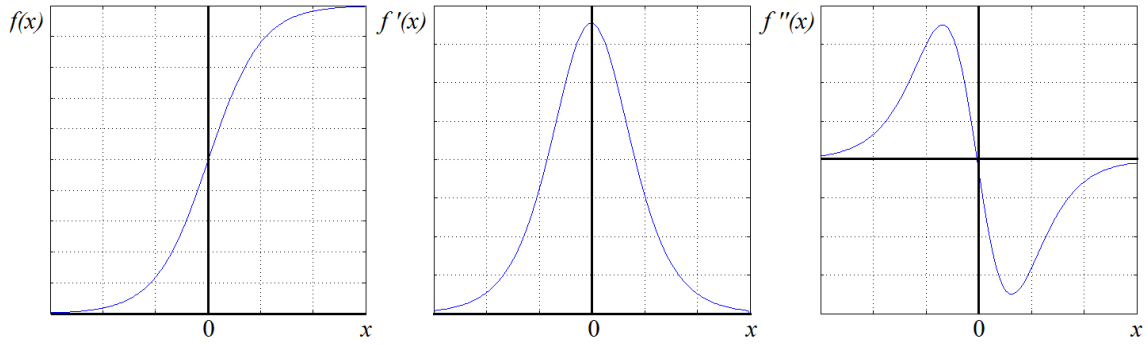
Robertsův:
$$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.5}$$

Sobelův:
$$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.6}$$

Prewittův:
$$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.7}$$

3.2.2 Zero crossing detekce hran

K nalezení informace, kde se nachází hrana, je možné využít i druhé derivace. Tento způsob je jednodušší, protože není hledán extrém, ale průchod nulou (viz *obr. 2*). V místech, kde dochází u obrazové funkce k největší změně intenzity, je první derivace největší a druhá derivace prochází nulou. Tuto metodu nelze použít, pokud je potřeba znát směr či velikost hran. Další nevýhodou může být dvojitá detekce některých hran nebo větší citlivost na šum [3].



Obr. 2 a) Původní obrazová funkce; b) první derivace funkce; c) druhá derivace funkce.

Druhou derivaci lze získat dvojnásobným zderivováním funkce. Rovnice pak bude mít tento tvar:

$$\Delta G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2} \tag{3.8}$$

I v tomto případě se nabízí možnost využít některý z hranových operátorů pro výpočet aproximovaných hodnot druhé derivace. Jedním z oblíbených operátorů je Laplacian-of-Gaussian (LoG), který kombinuje gaussovské vyhlazení obrazu a výpočet druhé derivace v jednom lineárním filtru. LoG operátor detekuje hrany s větší přesností než zmiňovaný Prewittův a Sobelův operátor [2].

3.2.3 Cannyho hranový detektor

Jedním z nejpopulárnějších a zároveň velice kvalitním hranovým detektorem je Cannyho hranový detektor. John F. Canny [4] si dal za cíl nelézt optimální algoritmus pro detekci hran. Definoval tři podmínky, které musí dokonalý detektor hran splňovat:

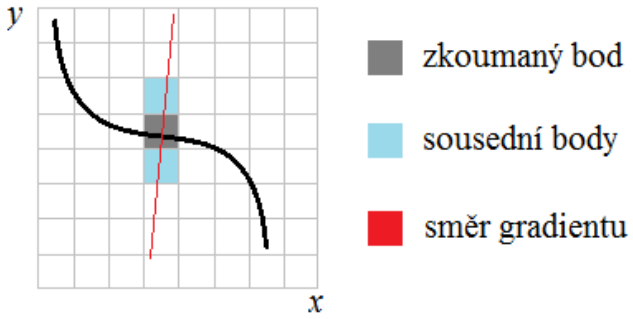
- *dobrá detekce* – algoritmus by měl detekovat co nejvíce reálných a minimum falešných hran

- *správná lokalizace* – detekovaná hrana by měla být, co nejblíže reálné hrany
- *minimální odezva* – každá hrana by měla být detekována jen jednou

Cannyho hranový detektor lze zařadit do gradientních metod (využívajících první derivaci), protože k výpočtu je za nutné znát velikost i směr gradientu. Samotný algoritmus pro detekci hran se skládá z několika kroků:

1. Redukce šumu
2. Výpočet amplitudy a směru gradientu intenzity
3. Ztenčení hran (non-maximum suppression)
4. Hysterezní prahování

V prvním kroku je eliminován šum rozmazáním obrazu. Toho lze dosáhnout opět konvolucí a to použitím Gaussova filtru jako konvoluční jádro. V druhém kroku se postupuje stejně, jako je popsáno v kapitole 2.2.1. Po této operaci jsou výsledkem detekované hrany s velkou tloušťkou, která je nežádoucí. Tyto tlusté hrany je potřeba ztenčit a také zjistit místo, kde se reálně v původním obraze vyskytují. Tento proces se nazývá non-maximum suppression. Pro každý bod jsou určeny sousední body ve směru gradientu. Má-li alespoň jeden ze sousedních bodů vetší intenzitu, než zkoumaný bod, je potlačen (přiřadí se zkoumanému bodu hodnota nula). Následující obrázek (o*br. 3*) znázorňuje výběr sousedních bodů.



Obr. 3 Non-maximum suppression.

Gradienty s velkou intenzitou budou pravděpodobněji hledané hrany než gradienty s nižší intenzitou, proto je potřeba v posledním kroku použít hysterezní prahování, aby byly eliminovány nežádoucí, případně falešné hrany. Hysterezní prahování je pouze vylepšení klasického prahování, namísto jednoho prahu využívá horní a dolní práh. Po dokončení tohoto procesu je výstupem binární obraz, kde každý je pixel označen 0 nebo 1, resp. je a není hrana [4][5].

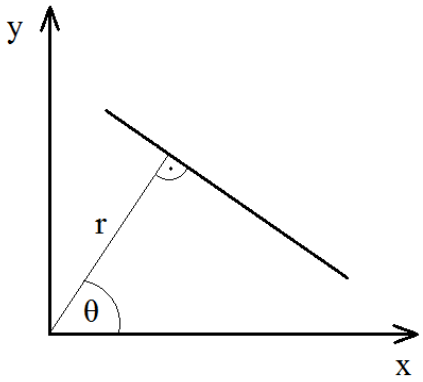
3.2.4 Houghova transformace

Tato metoda je používána k detekci objektů, u kterých lze hrany popsat jednoduchými křivkami, jako jsou přímky, kružnice, elipsy aj. Tato metoda detekce se vyznačuje vysokou odolností vůči šumu. Jedná se o transformaci kartézského souřadného systému do polárního. Princip fungování této metody je demonstrován na detekci přímky (viz *obr. 5*). Rovnice přímky je v prostoru dána vztahem:

$$x \cdot \cos\theta + y \cdot \sin\theta = r\,, \tag{3.9}$$

kde θ je úhel mezi normálou vedenou od středu souřadného systému k přímce a osou x, r je vzdálenost na této normále od středu souřadného systému k přímce (viz *obr. 4*).

Po dosazení souřadnic některého z bodů tvořících hranu do předchozí rovnice a vykreslení všech možných hodnot parametrů θ a r, vznikne v Houghově prostoru spojitá křivka. Pokud jsou vykresleny všechny body ležící na přímce, lze vypozorovat, že křivky se protínají v jednom bodě, který představuje parametry hledané přímky.

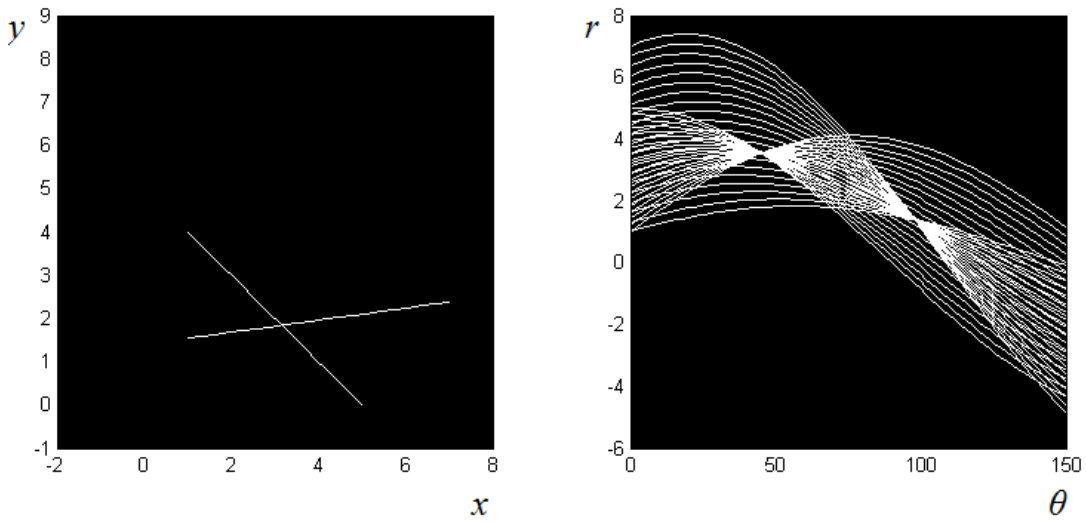


Obr. 4 Reprezentace přímky pomocí parametrů θ a r.

Tento postup lze využít i při hledání jiných objektů. Parametrické vyjádření kružnice by bylo:

$$(x-x_0)^2+(y-y_0)^2=r^2, \tag{3.10}$$

kde r, x_0, y_0 jsou hledané parametry. V tomto případě bude mít Houghův prostor tři dimenze a výpočet bude o to náročnější [6].



Obr. 5 Houghova transformace přímek vykreslená s využitím Matlabu.

3.3 Metody založené na detekci oblastí

Tyto metody detekují přímo oblasti v obrazu. V zašuměných scénách dokážou nalézt oblasti mnohem úspěšněji, než metody založené na detekci hran. Hlavním segmentačním kritériem pro detekci oblastí v obraze je homogenita oblastí. Kritérium homogenity může být založeno na několika vlastnostech, jako je úroveň šedi, sytost barvy, odstín barvy, textura, tvar apod.

3.3.1 Region growing

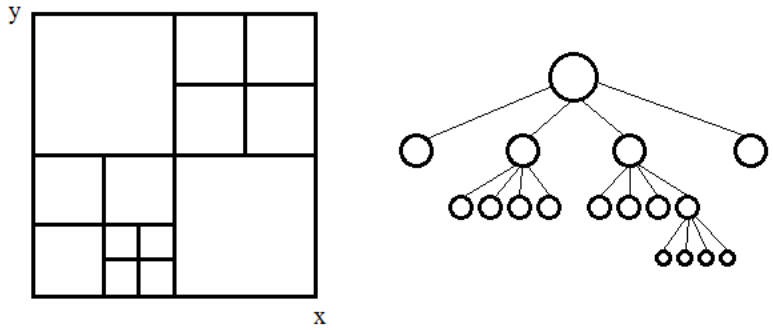
Metoda šíření oblastí je relativně stabilní způsob segmentace. Najde uplatnění v mnoha případech, především u segmentace zašuměných obrazů, protože tato metoda se na rozdíl od metod detekující přímo hrany daleko lépe vypořádává se šumem.

Region growing je založen na zjišťování podobnosti sousedních pixelů a celkové oblasti. Nové pixely jsou porovnávány s lokální průměrnou homogenitou oblasti. Pokud má zkoumaný pixel podobné vlastnosti jako jeho okolí, je označen jako patřící do oblasti. Tento proces probíhá do té doby, dokud nedojde k přiřazení všech pixelů do příslušné oblasti.

Většina algoritmů uplatňující region growing vychází z klasických SRG algoritmů [7] (Seed Region Grow). Tento algoritmus nejprve vybere několik počátečních bodů, ze kterých následně dochází k rozrůstání oblastí. Kvalita výsledku je závislá především na výběru počátečních bodů. Čím lépe budou počáteční body charakterizovat danou oblast, tím lepšího výsledku je dosaženo. Nejpoužívanější metody proto kombinují několik technik k tomu, aby dosáhly co nejlepších výsledků [8].

3.3.2 Split and merge

Tento algoritmus vyvinul T. Pavlidis v roce 1974 a stále je jeden z nejoblíbenějších a široce používaných k základní segmentaci obrazu. Obraz a jeho podobrazy jsou postupně děleny na čtyři kvadranty, pokud je oblast podle definovaných podmínek určena jako nehomogenní (viz *obr. 6*). Vzniklé sousední čtverce jsou naopak spojeny, pokud splňují podmínku homogenity.

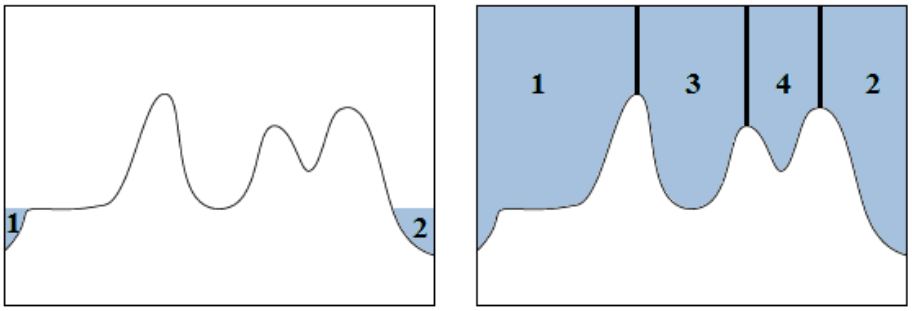


Obr. 6 Split and merge, quad-tree struktura.

Podmínka homogenity pro dělení oblastí nemusí být stejná jako podmínka určující spojovaní oblastí, avšak je potřeba nastavit podmínky tak, aby nedocházelo k zbytečnému nárůstu počtu iterací a tak ke zvýšení výpočetní náročnosti celého algoritmu [9].

3.3.3 Watershed

Watershed (česky rozvodí) je velice zajímavý způsob segmentace, jeho myšlenka pochází z geografie. Obrazová funkce je chápána jako reliéf krajiny. Počet segmentů je poté roven počtu nížin tj. počtu lokálních minim v obrazové funkci. Segmentace je tvořena tak, že z nížin stoupá hladina vody a zalévá okolní krajinu. V místech, kde by se mohla voda ze dvou různých povodí slít dohromady, jsou vytvořeny hráze (viz *obr. 7*). Zaplavování krajiny postupuje do té doby, kdy hladina vody dosáhne nejvyššího bodu reliéfu. Výsledná segmentace je tvořena jednotlivými regiony (povodími) oddělenými hrázemi.



Obr. 7 Průběh watershed segmentace [10].

Watershed je efektivní a osvědčená metoda segmentace. Její nevýhodou ovšem je, že produkuje nadměrný počet segmentů a to hlavně u zašuměných obrazů. Tento nedostatek se dá vyřešit použitím některého z algoritmů pro spojování oblastí nebo redukcí šumu původního obrazu [10].

3.4 Prahování

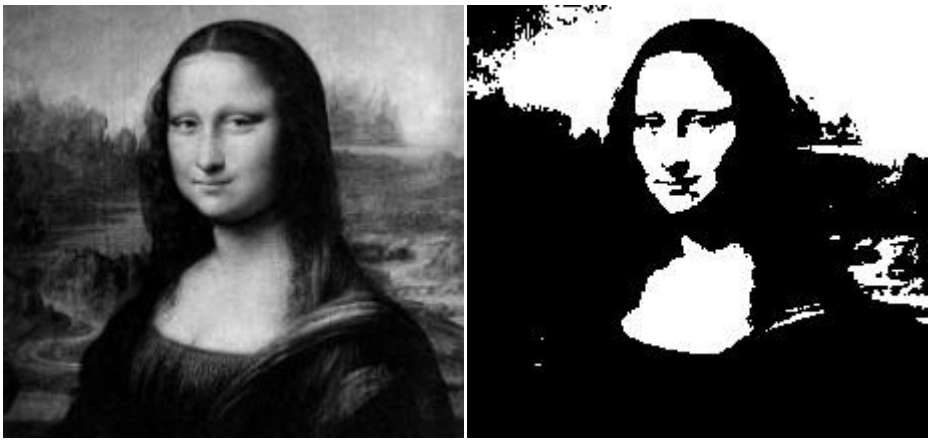
Prahování je nejjednodušší způsob segmentace. Využívá myšlenky, že objekty či zkoumané oblasti ve scéně mají stejnou úroveň intenzity a rozdílnou od intenzity pozadí. Prahování je nejstarší metoda segmentace obrazu, ale pro jednoduché případy se používá dodnes a to hlavně díky velmi malé výpočetní náročnosti. Také je součástí mnoha pokročilejších algoritmů pro zpracování obrazu. Ukázka prahování je zobrazena na *obr. 8*.

Prahování je transformace vstupního obrazu f na výstupní binární obraz g podle vztahu:

$$g(x,y)=\begin{cases}1 & pro\ f(x,y)\geq T\\ 0 & pro\ f(x,y)<T,\end{cases} \tag{3.11}$$

kde T je definovaná hodnota prahu. Funkce *g(x,y)* je rovna 1 pro body reprezentující objekty a 0 pro body spadající do pozadí (mají nižší intenzitu, než je definovaný práh).

Úspěšnost segmentace prahováním závisí především na hodnotě zvoleného prahu. V reálných případech není možné nastavit jeden práh pro celý obraz, použití globálního prahování většinou nepřinese požadovaný výsledek. Z toho důvodu je vhodnější nastavit hodnotu prahu zvlášť podle lokálních vlastností obrazu.



Obr. 8 Prahování obrazu Mony Lisy v odstínech šedi.

Jednou z modifikací prahování je *poloprahování*. Slouží k tomu, aby objekty byly odděleny od pozadí, ale zároveň se zachovala jasová informace na samotných objektech.

$$g(x,y)=\begin{cases}f(x,y) & pro\ f(x,y)\geq T\\ 0 & pro\ f(x,y)<T\end{cases} \tag{3.12}$$

Další modifikací je *prahování s více prahy*. Výsledkem už není binární obraz, ale obraz s menším počtem úrovní jasu.

$$g(x,y)=\begin{cases}1 & pro\ f(x,y)\in A_1\\ 2 & pro\ f(x,y)\in A_2\\ \vdots & \\ n & pro\ f(x,y)\in A_n\\ 0 & jinak,\end{cases} \tag{3.13}$$

kde $A_1, A_2, \ldots A_n$ jsou intervaly jednotlivých jasových úrovní [11].

4 VÝBĚR VHODNÉ METODY

Detekovat přímo obdélník je poměrně komplikovaná záležitost, nicméně pokud je na obdélník nahlíženo jako na čtyři přímky se specifickými geometrickými vlastnostmi, situace se značně zjednoduší. Celý modul pro zpracování obrazu bude využívat několik metod zpracování obrazu, jejich vhodná kombinace pak zajistí správnou detekci objektů.

Dobrý způsob jak nalézt přímku v obraze, je využít Houghovy transformace. Tento způsob je velice stabilní a dokáže detekovat přímky i ve velmi zašuměných obrazech. K tomu, aby byl detekován obdélník, není možné převést do Houghova prostoru celý obraz najednou. Výsledkem by byla změť bodů reprezentující detekované přímky, ze které by nebylo možné určit, jaké přímky tvoří jednotlivé hrany příslušných obdélníků. Z toho důvodu je Houghova transformace postupně prováděna ze všech možných výřezů původního obrazu, přičemž výřez má tvar čtverce o velikosti nejdelší hrany detekovaného obdélníku. Další nespornou výhodou je snadná úprava algoritmu pro případ, kdy by zkoumané objekty byly kružnice či elipsy.

Ovšem ještě předtím než bude moct být obraz převeden do Houghova prostoru, je nutné ze vstupního obrazu vytvořit mapu hran, binární obraz, kde 0 označuje pozadí a 1 hranu. K tomuto účelu byl vybrán Cannyho hranový detektor, především pro jeho jednoduchost a přesnost.

Výpočetní náročnost takového algoritmu by byla velká, protože by se Houghova transformace prováděla pro velké množství výřezů, které je úměrné počtu bodů v obraze. Počet výřezů se dá jednoduše snížit tím, že algoritmus bude zkoumat pouze ty oblasti, kde dané objekty leží. Toho lze dosáhnout vhodným prahováním vstupního obrazu, tím oddělit objekty od pozadí a Houghovu transformaci provádět pouze pro místa, kde objekty opravdu leží.

Výběr této metody usnadnil také fakt, že při detekci pracuje přímo s parametry popisujícími vlastnosti objektů. Díky tomu odpadá podstatná část výpočtů, které by dodatečně zjišťovaly rozměry, střed a natočení objektů.

5 IMPLEMENTACE

Celý proces začínaje nasnímáním pracovní plochy a přesunem objektů konče, lze rozdělit do několika základních kroků. Jednotlivé kroky budou v následujících podkapitolách popsány podrobněji.

Hlavní algoritmus

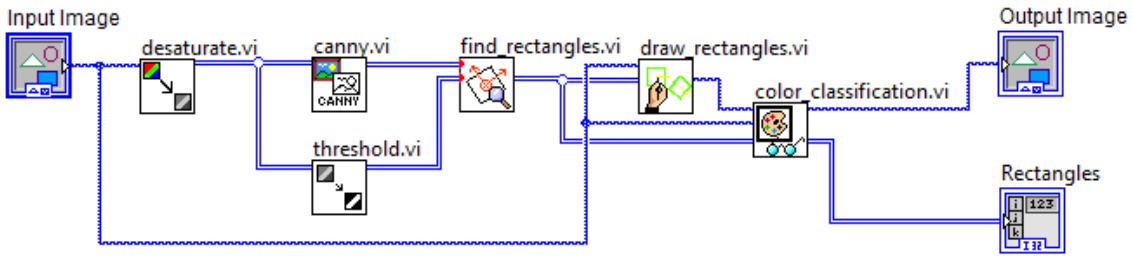
```
1:  Img = GetWebcamPicture                    //načte obraz z webkamery
2:  Rectangles = ImageProcess(Img)            //vrátí pole s polohou, rozměrem, natočením a
                                                barvou detekovaných objektů
3:  for all detected rectangles do
4:      Servo = Kinematics(Rectangles)        //vrátí pole obsahující úhly natočení
                                                jednotlivých serv
5:      BuildSendPackets(Servo)               //sestaví příkazové pakety a odešle je přes
                                                COM port do mikrokontroléru
6:  end for
```

5.1 Implementace zvolené metody ke zpracování obrazu

Algoritmus pro zpracování obrazu je nejdůležitější částí programu. Jeho vstupem je nasnímaný obraz z webkamery a výstupem pole obsahující parametry (pozice, natočení, barva) detekovaných objektů. Pro názornost detekce je součástí výstupu i obrázek s vykreslenými detekovanými objekty. Schéma algoritmu v prostředí LabVIEW je zobrazeno na *obr. 9*.

Algoritmus ImageProcess

```
1:  ImageProcess(Img)
2:      GrayImg = Desaturate(Img)                                   //odbarví vstupní obraz
3:      EdgeMap = Canny(GrayImg)                                    //vytvoří mapu hran
4:      SearchMask = TresholdErosion(GrayImg)                       //vytvoří binární masku
5:      Rectangles = FindRectangles(EdgeMap, SearchMask)            //vrátí pole s parametry
                                                                      detekovaných obdélníků
6:      ImgOut = DrawRectangles (Img, Rectangles)                   //vykreslí detekované
                                                                      objekty do obrazu Img
7:      ImgOut, Rectangles = ColorClassification(ImgOut, Img, Rectangles) //zjistí barvu objektů
                                                                            a do pole Rectangles
                                                                            přidá informaci o barvě
8:  return ImgOut, Rectangles.
```



Obr. 9 Schéma zpracování obrazu v prostředí NI LabVIEW.

5.1.1 Vytvoření mapy hran

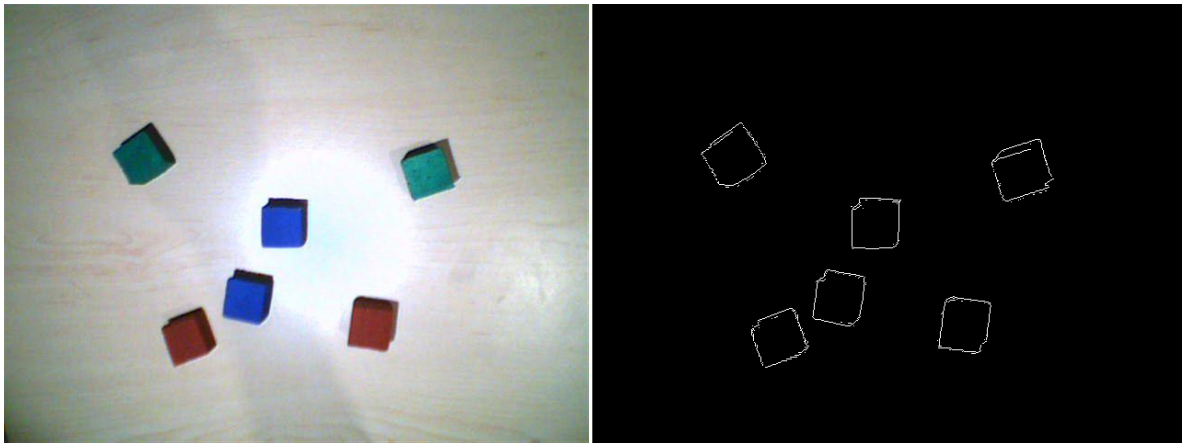
Prvním krokem je vytvoření mapy hran z původního obrazu. Tato mapa hran (*obr. 10b*) pak bude sloužit jako vstupní obraz pro Houghovu transformaci. K detekci hran je použit Cannyho hranový detektor, ale ještě předtím je nutné převést obraz do odstínů šedi.

Algoritmus Desaturate

```
1:  Desaturate(Img)
2:      for all image coordinates (x,y) do
3:          Red = GetRedValue(x,y)                    //zjistí velikost červené složky pixelu
4:          Green = GetGreenValue(x,y)                //zjistí velikost zelené složky pixelu
5:          Blue = GetBlueValue(x,y)                  //zjistí velikost modré složky pixelu
6:          GrayImg[x,y] = Red · 0.299 + Green · 0.587 + Blue · 0.114   //vrátí odbarvený pixel
7:      end for
8:  return GrayImg.
```

Algoritmus Canny

```
1:  Canny(GrayImg)
2:      GrayImg = Convolution(GrayImg,GaussMask)   //rozmaže obraz Gaussovským filtrem
3:      Gx = Convolution(GrayImg, SobelX)          //vypočte aproximovanou hodnotu
                                                     první derivace podle osy x
4:      Gy = Convolution(GrayImg, SobelY)          //vypočte aproximovanou hodnotu
                                                     první derivace podle osy y
5:      G = sqrt(Gx · Gx + Gy · Gy)                //vypočte velikost gradientu
6:      GAng = atan(Gy/Gx)                         //vypočte směr gradientu
7:      GrayImg = NMS(G,GAng)                      //non maximum suppression
8:      EdgeMap = Threshold(GrayImg)               //vytvoří mapu hran
9:  return EdgeMap.
```



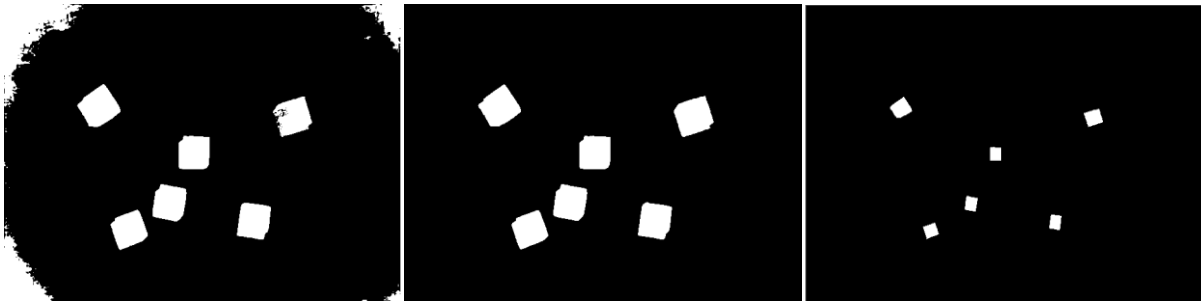
Obr. 10 a) Původní obraz; b) mapa hran.

5.1.2 Vytvoření binární masky

Dalším krokem je vytvoření binární masky, která bude sloužit k ověření, zdali se ve zkoumaném výřezu nachází střed obdélníku. Objekty na podložce mají výrazně tmavší odstín než podložka, proto bylo použito prahování, jedna z nejjednodušších segmentačních technik, nicméně v tomto případě zcela postačující. Použití globálního prahu není kvůli nerovnoměrnému osvětlení plochy možné, došlo by tak k označení tmavých rohů jako objekty. Řešením je rozdělení obrazu na menší části a pro každou část provést prahování zvlášť (viz *obr. 11b*). Obraz byl rozdělen na 12 částí (4x3), pro každou oblast byla vypočítána průměrná hodnota intenzity, hodnota prahu je potom rovna 65% průměrné hodnoty intenzity. Jelikož stačí znát pouze oblast, kde se nachází střed obdélníku, není nutné prohledávat celou plochu objektu, je tedy možné zmenšit nalezené segmenty. Toho lze dosáhnout užitím eroze (viz *obr. 11c*), jednoho ze základních prvků matematické morfologie.

Algoritmus ThresholdErosion

```
1:   ThresholdErosion(Img)
2:       Tiles = Subdivide(Img,4,3);          //rozdělí obraz na 4x3 menších oblastí
3:       for all tiles do
4:           Average = AverageValueOfArray(Tiles)   //pro každou podoblast zvlášť se
                                                    vypočte průměrná hodnota intenzity
5:           ThreshValue = 0.65 · Average
6:           Tiles = Threshold(Img,ThresValue)      //provede prahování podoblastí
                                                    s prahem 65% průměrné hodnoty
                                                    intenzity dané podoblasti
7:       end for
8:       Img = Merge(Tiles)                   //spojí zpět jednotlivé podoblasti do
                                              původního obrazu
9:       SearchMask = Erosion(Img,15)         //15x zmenši nalezené segmenty
10:  return SearchMask.
```



Obr. 11 a) Prahování s globálním prahem; b) prahování s rozdělením do podoblastí; c) eroze.

5.1.3 Rozpoznávání obdélníků

Jak již bylo řečeno, hlavním prvkem modulu pro zpracování obrazu je Houghova transformace. Přímka je ve 2D prostoru dána rovnicí:

$$x \cdot \cos\theta + y \cdot \sin\theta = r \tag{5.1}$$

Za předpokladu, že počátek souřadného systému obrazu, ze kterého je prováděna Houghova transformace, je zároveň střed obdélníku (viz *Obr. 12a*), mají poté nalezené body v Houghově prostoru velmi specifické vlastnosti. Protilehlé strany obdélníka mají stejnou velikost vzdálenosti *r* a zároveň se jejich úhly θ liší o 180°.

$$\Delta r = \left| r_i + r_j \right| \le T_r \tag{5.2}$$

$$\Delta\theta = \left| \theta_i + \theta_j \right| \le T_\theta \tag{5.3}$$

kde T_r a T_θ jsou prahy, hodnoty blízké nule, které ponechají určitou volnost v detekci. Pokud jsou tyto podmínky splněny, nalezené body v Houghově prostoru označují dvě rovnoběžné přímky. Přímky splňující podmínky jsou spárovány a uloženy jako $P_{1\ldots n} = (R, \Theta)$. Parametry R a Θ jsou průměry ze dvou původních hodnot r_i, r_j a θ_i, θ_j.

Nyní už jen stačí porovnat nalezené páry, pokud některý pár mezi sebou svírá úhel 90°, jedná se o obdélník.

$$\Delta\Theta = \left| \left| \Theta_i + \Theta_j \right| - 90° \right| \le T_\Theta \tag{5.4}$$

Velkou výhodou je, že pokud dojde k detekci obdélníku, automaticky je zjištěn jeho střed (počátek souřadného systému z původního obrazu) a i natočení (úhel svírající normála jedné ze stran obdélníku s osou x souřadného systému). K tomu, aby bylo zajištěno, že střed obrazu, ze kterého se provádí Houghova transformace, náleží zároveň středu obdélníku, je nutné skenovat celý obraz čtvercovým výřezem o délce strany rovnající se největšímu rozměru detekovaného objektu. Protože by výpočetní

náročnost této operace byla velmi vysoká, skenování probíhá na vytvořené binární masce. Pokud má pixel ve středu výřezu hodnotu 1, provede se Houghova transformace, ovšem nikoli z binární masky, ale z mapy hran.

Dále je nutné určit barvu detekovaného objektu. Snímané objekty budou mít buď červenou, zelenou nebo modrou barvu, to jsou přímo složky barevného formátu RGB. Bude tedy stačit vypočítat průměrné hodnoty jednotlivých barevných složek. O výsledné barvě objektu poté rozhodne největší hodnota z vypočítaných průměrů. V případě, kdy by bylo potřeba rozlišovat více barev, bylo by vhodnější převést barvy do formátu HSL. Pro jednotlivé barvy by pak byl přiřazen určitý rozsah hodnot složky hue.

Jelikož v reálné situaci nemají detekované objekty dokonalý obdélníkový tvar, byla ponechána určitá volnost v parametrech určující jejich tvar. Z toho důvodu může v mnoha případech docházet k vícenásobné detekci jednoho objektu. Tento nedostatek je vyřešen zprůměrováním parametrů obdélníků u objektů, které mají podobné souřadnice středu.

Po úspěšné detekci je možné do původního obrazu vykreslit nalezené objekty (viz *obr. 13*). Souřadnice vrcholů jednotlivých obdélníků byly získány výpočtem průsečíků stran, které tvoří obdélník. Tento krok není pro manipulaci objektů nutný a slouží jen k názorné ukázce detekce.

Algoritmus FindRectangles

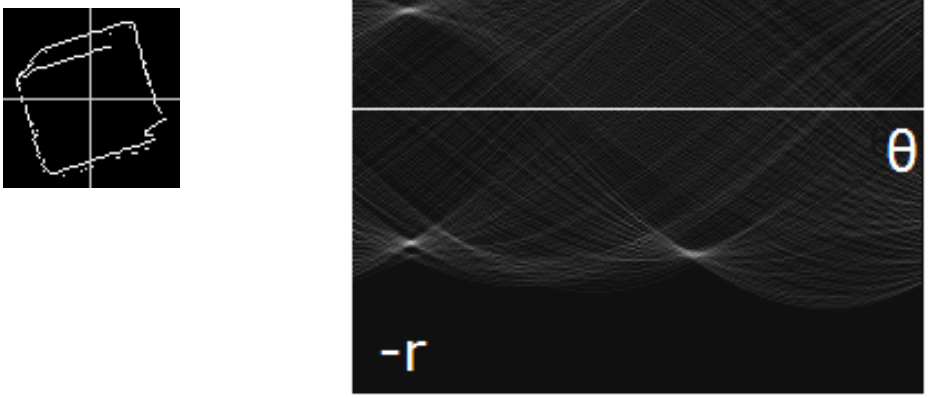
```
1:   FindRectangles(EdgeMap,SearchMask)
2:       SectionSize = 80;                          //velikost výřezu je volena dle
                                                      velikosti detekovaných objektů
3:       for all posible sections of the image (i,j) do   //skenuje celý obraz výřezem
                                                      o velikosti 80x80 pixelů
4:       Let (ic, jc) be the center coordinates of the section (i,j)
5:           if SearchMask[ic,jc] == 1 then         //kontroluje, zdali střed výřezu
                                                      leží na středu objektu
6:               ImgSection = ArraySubset(EdgeMap,i,j,SectionSize)   //vytvoří výřez
7:               HoughImg = HoughTransform(ImgSection)   //provede Houghovu
                                                      transformaci výřezu
8:               Rectangles = ProcessHough(HoughImg,ic,jc,Rectangles)
                   //Z nalezených maxim vypočítá parametry detekovaných přímek. Pokud některé
                     z přímek tvoří obdélník, přidá do pole Rectangles řádek. Pole Rectangles tvoří
                     tyto sloupce: střed x, střed y, délka stran a, délka stran b, natočení stran a,
                     natočení stran b.
9:           end if
10:      end for
11:      Rectangles = RemoveDuplicates(Rectangles)   //eliminuje duplikátní detekce
12:  return Rectangles.
```

Algoritmus HoughTransform [2]

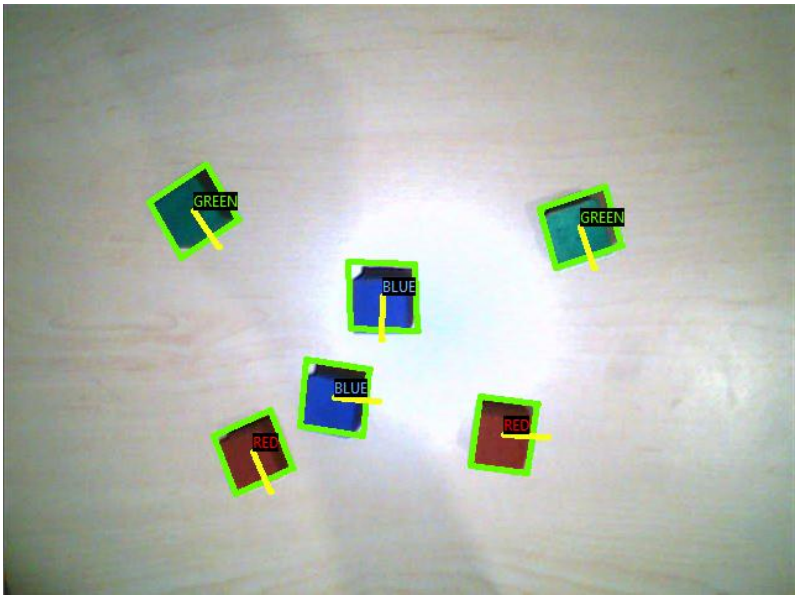
```
1:   HoughTransform(ImgSection)
2:       Initialize 2D accumulator array A(r,θ) to all zeros   //vytvoří pole Houghova prostoru
3:       Let (uc, vc) be the center coordinates of the ImgSection
4:       for all ImgSection coordinates (u,v) do
5:           if ImgSection[u,v] is an edge point then   //provede pouze pro body tvořící hranu
6:               x = u – uc                             //relativní vzdálenost x od středu obrazu
7:               y = v – vc                             //relativní vzdálenost y od středu obrazu
8:               for θ = 0 to π do
9:                   r = x · cos(θ) + y · sin(θ)        //rovnice přímky
10:                  A[r,θ]++                           //navýší hodnotu daného prvku pole A o 1
11:              end for
12:          end if
13:      end for
14:      HoughImg = A
15:  return HoughImg.
```

Algoritmus ProcessHough

```
1:    ProcessHough(HoughImg,ic,jc,Rectangles)
2:        Initialize empty 2D array P
3:        M = HoughPeaks(HoughImg)                    //zjistí maxima a vytvoří pole bodů M(r, θ)
4:        for all pairs of M points (r_i ,r_j and θ_i, θ_j) do    //nalezne páry rovnoběžných přímek
5:            Δr = abs(r_i + r_j)
6:            Δθ = abs(θ_i – θ_j)
7:            if Δr ≤ T_r and Δθ ≤ T_θ then
8:                R = r_i + r_j
9:                Θ = (θ_i + θ_j) / 2
10:               Add row (R,Θ) to array P
11:           end if
12:       end for
13:       for all pairs of P lines (R_i,R_j and Θ_i,Θ_j) do    //zjistí, zda některé přímky tvoří obdélník
14:           ΔΘ =  abs(abs(Θ_i – Θ_j) – 90°)
15:           if ΔΘ ≤ T_Θ then
16:               Add row (ic,jc,R_i,R_j,Θ_i,Θ_j) to array Rectangles
17:           end if
18:       end for
19:   return Rectangles.
```



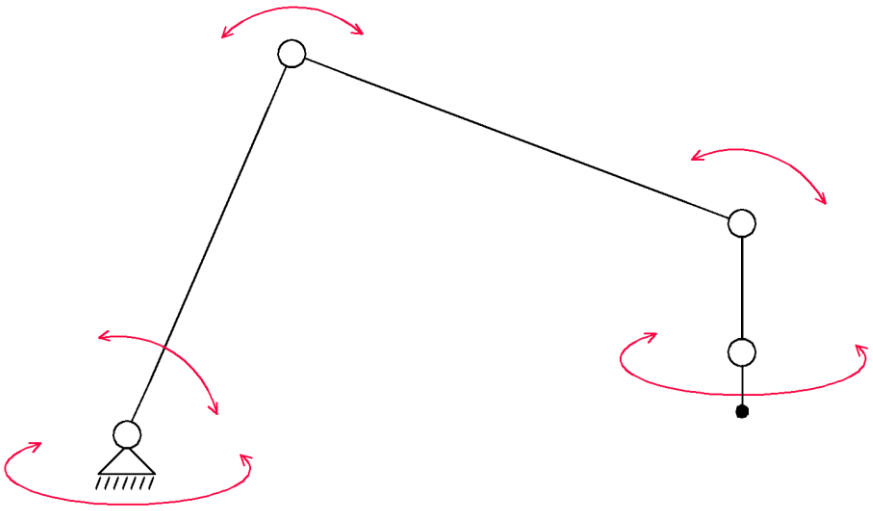
Obr. 12 a) Zkoumaný výřez; b) Houghova transformace výřezu.



Obr. 13 Detekované objekty.

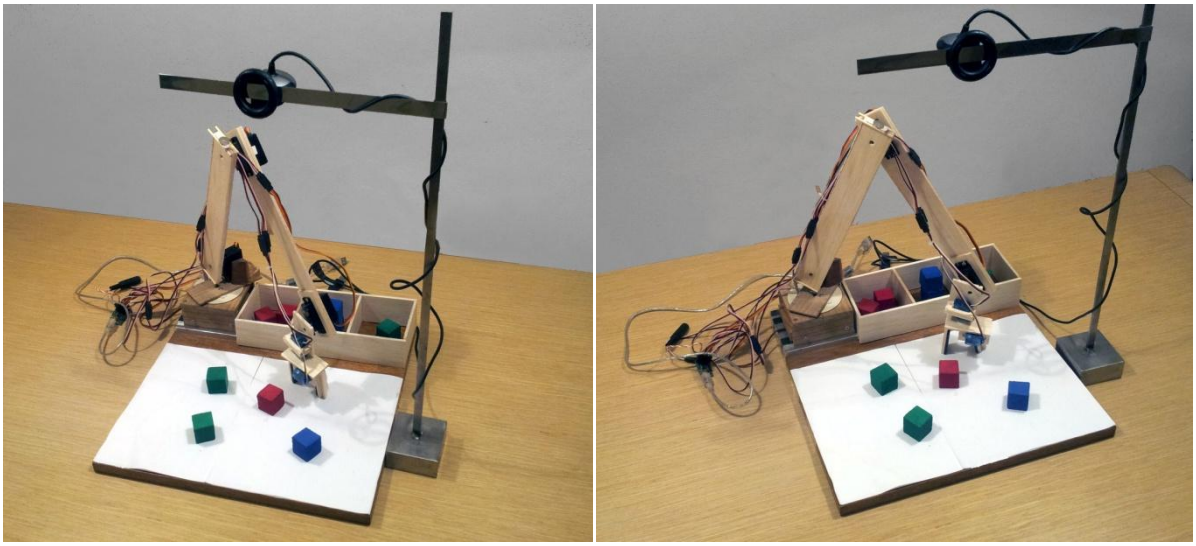
5.2 Konstrukce robotického manipulátoru

Ze dvou možných konstrukčních řešení robotického manipulátoru byl upřednostněn typ využívající sériovou kinematikou před kinematikou paralelní. Bylo tak voleno především z důvodu jednoduché konstrukce a nízkých pořizovacích nákladů. Objekty určené k manipulaci budou ve všech případech sbírány směrem kolmo od podložky, z toho vyplývá, že postačí manipulátor s pěti stupni volnosti (5 DOF). Kinematické schéma manipulátoru lze vidět níže na *obr. 14*.



Obr. 14 Schéma manipulátoru.

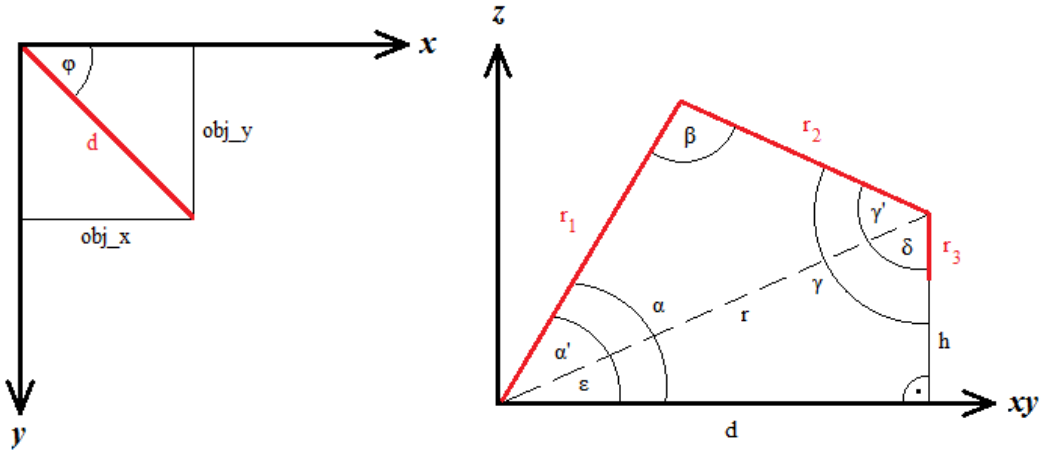
Použitým konstrukčním materiálem, mezi modeláři hojně využívaným, je balzové dřevo. Hlavní výhodou tohoto materiálu je velmi nízká hmotnost při zachování dostatečné pevnosti. Dalším plusem je snadná opracovatelnost. Rotace v kloubech je zajištěna pomocí čtyř serv Tower Pro MG995 s kovovými převody určenými pro vysokou zátěž. Pro natočení a svírání uchopovacího mechanismu byla použita dvě podstatně slabší serva Micro 9g. Další součástí zařízení je stojan s webkamerou Microsoft LifeCam VX-3000, která je fixně nastavena tak, aby kolmo snímala pracovní plochu vytyčenou dřevotřískovou deskou. Výsledná podoba celého zařízení je zobrazena na *obr. 15*.



Obr. 15 Fotografie modelu robotického manipulátoru.

5.3 Kinematika robotického manipulátoru

K tomu, aby se konec robotického ramene dostal nad požadovanou pozici, je nutné vypočítat úhly, o které se mají natočit jednotlivá serva. Vstupními parametry výpočtu jsou osové rozteče mezi klouby ramen, vzdálenost koncového bodu uchopovacího mechanismu od podložky, souřadnice a natočení objektů.



Obr. 16 a) Pohled shora; b) pohled z boku.

Výpočet úhlu natočení celého ramene nad objekt lze snadno vyvodit z předchozího nákresu (*obr. 16*). Úhel φ pak odpovídá natočení prvního serva. Vzdálenosti určující polohu objektu obj_x a obj_y jsou výsledkem zpracování obrazu. Pro následující výpočty je také potřeba vypočítat vzdálenost objektu od osy manipulátoru d.

$$\varphi = \arcsin\frac{obj\_y}{obj\_x} \tag{5.5}$$

$$d = \sqrt{obj\_x^2 + obj\_y^2} \tag{5.6}$$

Ani úhly natočení zbývajících serv nebudou na výpočet nikterak složité. Úhel α odpovídá úhlu natočení druhému servu, úhel β třetímu a úhel γ servu čtvrtému.

$$r = \sqrt{d^2 + \left(h + r_3\right)^2} \tag{5.7}$$

$$\alpha = \alpha' + \varepsilon = \arccos\frac{r^2 + r_1^2 - r_2^2}{2 \cdot r \cdot r_1} + \arcsin\frac{h + r_2}{d} \tag{5.8}$$

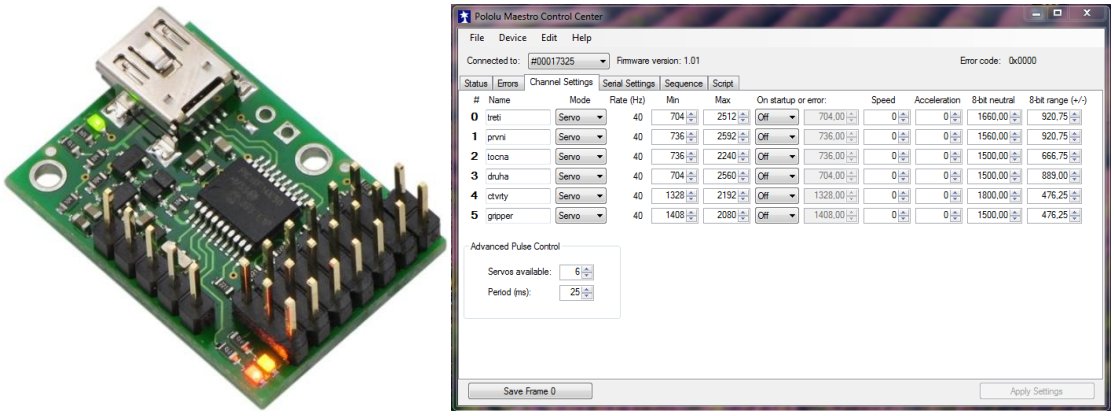
$$\beta = \arccos\frac{r_2^2 + r_1^2 - r^2}{2 \cdot r_2 \cdot r_1} \tag{5.9}$$

$$\gamma = \gamma' + \delta = \arccos\frac{r_2^2 + r^2 - r_1^2}{2 \cdot r_2 \cdot r} + (180° - 90° - \varepsilon) \tag{5.10}$$

Poslední servo, které zajišťuje natočení uchopovacího mechanismu, se natočí o úhel rovnající se součtu úhlu φ a úhlu odpovídající natočení objektu, ten je získán zpracováním obrazu.

5.4 Řízení robotického manipulátoru

K řízení servomotorků jsem se rozhodl využít jedno z existujících řešení, konkrétně šestikanálový mikrokontrolér Micro Maesto od firmy Pololu. Micro Maestro. Umožňuje řídit až 6 serv nezávisle na sobě, pro každé z nich lze nastavit rychlost i zrychlení. Se zařízením je dodáván software Pololu Maestro Control Center (viz *obr. 17*), ten je schopen serva přímo ovládat, ovšem v mém případě posloužil pouze ke kalibraci serv (nalezení středu a zajištění maximálního rozsahu).



Obr. 17 Pololu Micro Maestro 6-channel a Pololu Maestro Control Center [12].

Mikrokontrolér Micro Maestro je řízen sériovými příkazy přes USB rozhraní. Komunikace je dosažena zasíláním příkazových paketů skládajících se z jednotlivých příkazových bajtů. Po nich následují datové bajty, které přísluší danému příkazu. Příkazové bajty mají vždy nejvýznamnější bity (MSB – most significant bit) s hodnotou 1, zatímco datové bajty mají nejvýznamnější bity s hodnotou 0. Z toho vyplývá, že každý data bajt může přenést pouze 7 bitů informací. Výjimkou je Mini SCC protokol, kde datové bajty mohou mít jakoukoliv hodnotu od 0 do 254. Maestro podporuje následující tři protokoly.

Compact Protocol – jednoduchý a kompaktní protokol, který se využívá, je-li Maestro jediné zařízení připojené na sériovou linku. Příkazový paket vypadá takto:

příkazový bajt (s MSB nastaveným na 1), **datové bajty**

Příklad zápisu v hexadecimálním tvaru je **0x84, 0x00, 0x70, 0x2E**. Příkazový bajt 0x84 značí příkaz pro nastavení pozice serva. První data bajt 0x00 udává pořadové číslo serva a zbylé dva data bajty určují pozici v jednotkách čtvrtiny mikrosekund.

Pololu Protocol – tento protokol je kompatibilní s dalšími zařízeními od Pololu a je využíván zejména při sériovém zapojení více zařizení. Příkazový paket má následující tvar:

0xAA, **číslo zařízení, příkazový bajt** (s MSB nastaveným na 0), **datové bajty**

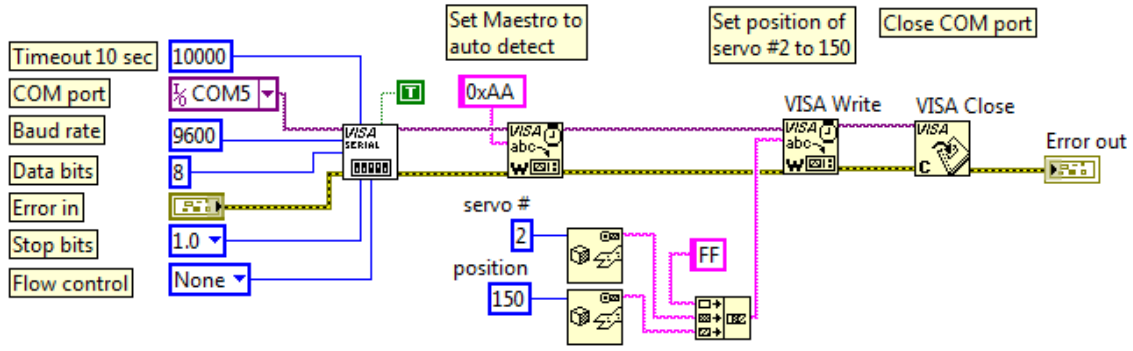
Příklad zápisu v hexadecimálním tvaru je **0xAA, 0x0C, 0x04, 0x00, 0x70, 0x2E.** Tento příkaz nastaví cílovou polohu serva s pořadovým číslem 0 na hodnotu 1500 µs na zařízení číslo 12.

Mini SSC Protocol – tento protokol je vhodný pro nastavení pozic jednotlivých serv. Zabere pouze tři bajty a proto je vhodný na rozesílání většího množství příkazů najednou. Příkazový paket potom vypadá takto:

0xFF, číslo serva, cílová pozice serva

Příklad zápisu v hexadecimálním tvaru je **0xFF, 0x00, 0x7F.** Tento příkaz nastaví servo s pořadovým číslem 0 na neutrální pozici (127) [12].

Všechny tři protokoly lze libovolně kombinovat a používat všechny příkazy dohromady zcela bez omezení. V mém případě budu využívat příkazy Set Speed (Compact Protocol) a Set Target (Mini SSC Protocol). Příkazové pakety budou vytvářeny a rovněž odesílány přes COM port pomocí NI LabVIEW (viz *obr. 18*).



Obr. 18 Ukázka nastavení polohy serva přes Mini SSC Protocol.

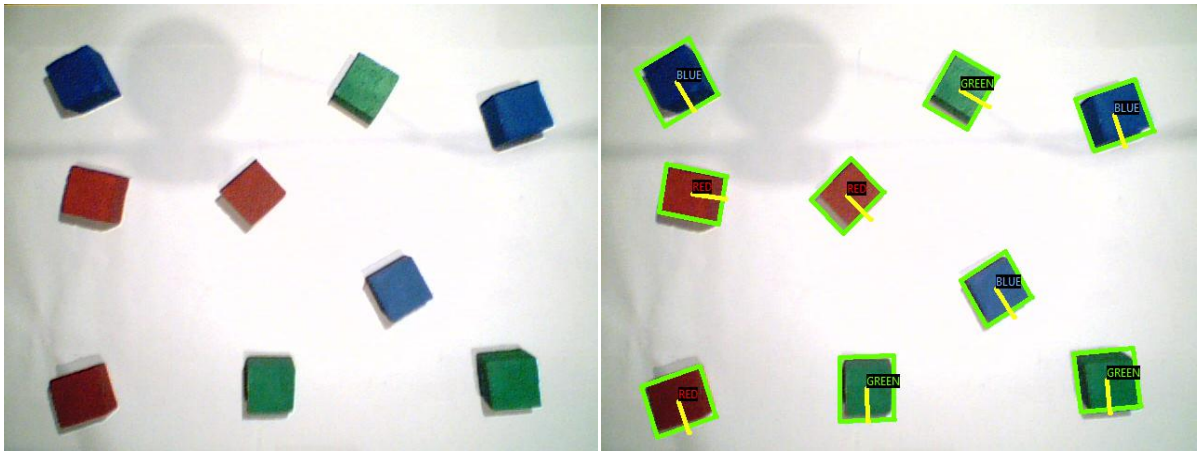
6 EXPERIMENT

6.1 Popis experimentu

Cílem experimentu je prověřit funkčnost a spolehlivost vybrané metody zpracování obrazu. Úspěšnost manipulace pak bude záviset především na správné detekci. Bylo realizováno několik modelových situací rozmístění objektů na pracovní ploše, při kterých bylo sledováno několik faktorů, jako jsou správná detekce středů, velikostí, natočení, barev objektů a dále také zda došlo k úspěšné manipulaci. Pro názornost byly vykresleny do původního obrazu detekované objekty.

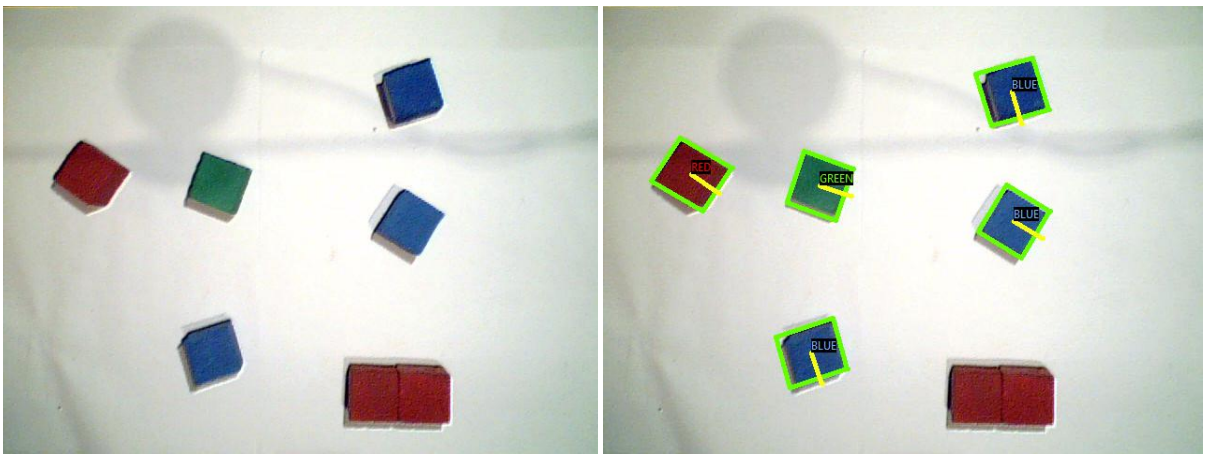
6.2 Výsledky

V první situaci byly kostky náhodně rozmístěny po pracovní ploše (viz *obr. 19*). Vzdálenost mezi jednotlivými objekty byla dostatečně velká k tomu, aby došlo k úspěšnému uchopení objektu. Detekce objektů byla úspěšná ve všech faktorech, následná manipulace proběhla též bez obtíží.



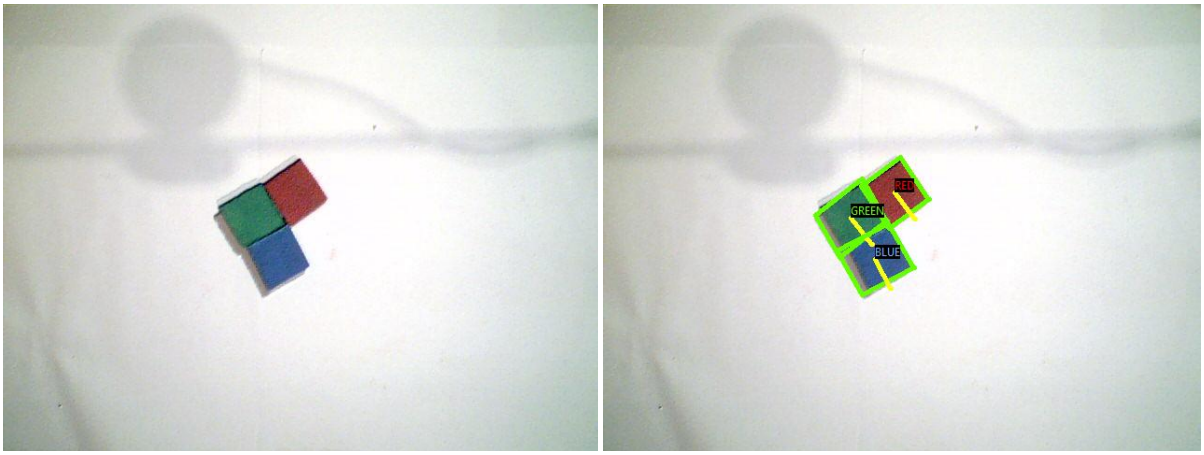
Obr. 19 Situace I., jednoduché rozmístění.

Druhý pokus se od prvního liší v umístění dvou červených kostek do těsné blízkosti (viz *obr. 20*). Tato situace měla modelovat objekt, který má větší rozměry než je stanovený limit. Po zpracování obrazu byl tento velký objekt správně neoznačen. V ostatních případech došlo k úspěšné detekci i manipulaci.



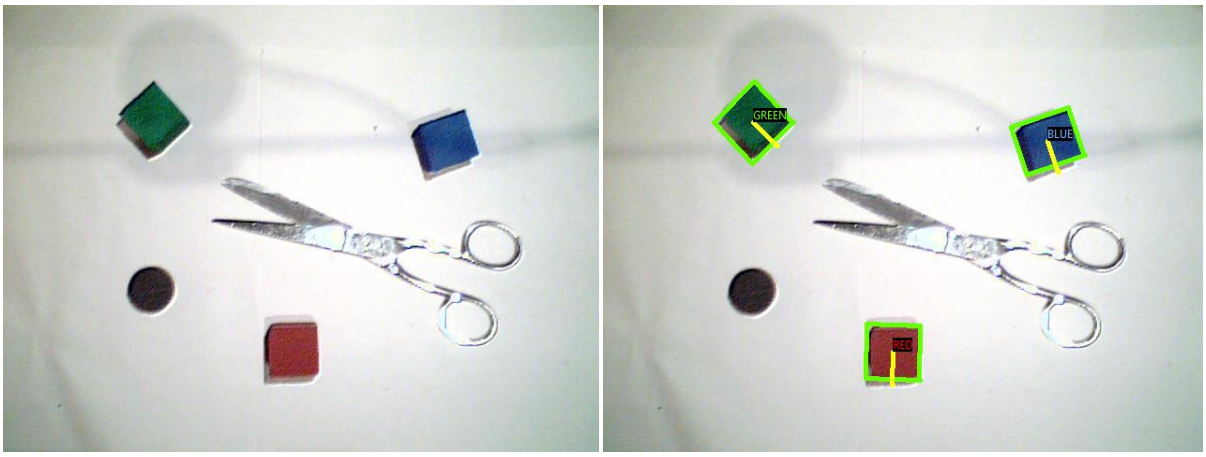
Obr. 20 Situace II., objekt přesahující rozměrový limit.

Ve třetí situaci byly umístěny tři objekty různé barvy těsně vedle sebe (viz *obr. 21*). Jelikož konstrukce manipulátoru nedovoluje uchopit objekt, který nemá kolem sebe dostatek prostoru, je zhodnocena pouze úspěšnost detekce. Vzhledem k situaci je možné považovat detekci za úspěšnou, nicméně přesnost byla v tomto případě nižší než v situacích předchozích.



Obr. 21 Situace III., tři objekty různé barvy umístěné v těsné blízkosti.

V posledním pokusu byly na pracovní plochu přidány objekty neobdélníkového tvaru, kulatý magnet a nůžky (viz *obr. 22*). Cílem bylo prověřit, zda opravdu dochází k detekci jen obdélníkových objektů. Magnet i nůžky byly úspěšně vyhodnoceny, jako objekty, které neslouží k manipulaci. U ostatních objektů došlo k úspěšné detekci i manipulaci.



Obr. 22 Situace IV., objekty neurčené k manipulaci.

Ve většině případů došlo k úspěšné a přesné detekci objektů, tam kde to bylo možné, byly všechny objekty i zdárně přesunuty do kontejnerů podle barev. Jediným vypozorovaným nedostatkem bylo detekování ostrých stínů, jako součást objektu. Tento neduh by bylo možné vyřešit vhodnějším nasvícením pracovní plochy.

7 ZÁVĚR

V této práci bylo cílem seznámit se s principy detekce objektů v obraze a poté navrhnout vhodnou metodu pro rozpoznání obdélníkových tvarů. Vybraná metoda byla využita při experimentu, který měl za úkol simulovat jedno z reálných využití zpracování obrazu. Před řešením daného problému bylo nutné blíže porozumět základním pojmům a prostudovat již existující techniky zpracování obrazu. Těmto teoretickým poznatkům se věnuje první část práce. Stručně jsou popsány základní přístupy ke zpracování obrazu, bylo poukázáno na jejich výhody, nevýhody a také pro jaké situace jsou vhodné.

V další části práce jsou aplikovány poznatky z teoretické části k vytvoření vlastního modulu pro zpracování obrazu. Dále je věnována pozornost konstrukci, kinematice a řízení robotického manipulátoru. Vyrobený model manipulátoru, kvůli méně kvalitním servomotorkům a pružné konstrukci z balzového dřeva, nevykazoval takovou přesnost jako skutečné průmyslové manipulátory, ovšem nepřesnosti se pohybovaly v řádech milimetrů, což je vzhledem k velikosti objektů zanedbatelná odchylka. Nutno konstatovat, že pokud byly objekty správně detekovány a měly kolem sebe dostatečně volného prostoru, proběhla manipulace ve všech případech bez problémů. Hlavní princip fungování modulu pro zpracování obrazu je zevrubně vysvětlen pomocí jednotlivých algoritmů, ze kterých je celý modul složen. Kombinace Cannyho hranového detektoru a Houghovy transformace se osvědčila jako velice spolehlivý nástroj k detekci jednoduchých tvarů s lehce definovatelnou geometrií. Za velký nedostatek považuji výpočetní náročnost této metody. Nutnost skenovat velkou část obrazu výřezem, pro každý výřez provádět Houghovu transformaci a tu následně zpracovat, si vyžaduje velký výpočetní výkon. V tomto směru se do budoucna naskýtá možnost lépe optimalizovat nebo vylepšit tuto metodu. Naopak velkou výhodou je spolehlivost, přesnost a jednoduchá modifikovatelnost, díky které by mohlo být uplatnění této metody velice široké.

Experiment probíhal se čtyřmi modelovými situacemi rozmístění objektů. Jak již bylo řečeno, úspěšnost robotické manipulace objektů byla závislá hlavně na úspěšnosti rozpoznávání objektů. Z toho důvodu byla během experimentu sledována zejména kvalita detekce. Výsledek experimentu jen potvrdil úspěšnost detekce a schopnost vypořádat se s různými případy rozmístění objektů.

Do budoucna bych se rád zabýval optimalizací, nalezením dalšího uplatnění této metody detekce a také jejím vylepšením. Výpočty využívané ke zpracování 2D obrazu většinou spočívají v jednoduchých, ale dosti početných operacích s dvourozměrnými poli. Všechny tyto výpočty byly realizovány přímo ve vývojovém prostředí NI LabVIEW. Ukázalo se, že velice vhodné by bylo využít DLL knihovny, které by značně zjednodušily kód a také zvýšily výpočetní rychlost celého programu.

8 SEZNAM POUŽITÉ LITERATURY

[1] FIŘT, Jaroslav a Radek HOLOTA. *Digitalizace a zpracování obrazu* [online]. [cit. 2013-02-19]. Dostupné z: http://home.zcu.cz/~holota5/publ/DigZprO.pdf

[2] BURGER, Wilhelm. *Digital image processing: an algorithmic introduction using Java*. 1st ed. New York: Springer, 2008, xx, 564 s. ISBN 978-1-84628-379-6.

[3] ŠPANĚL, Michal. *Obrazové segmentační techniky: Přehled existujících metod* [online]. 2005 [cit. 2013-02-16]. Dostupné z: http://www.fit.vutbr.cz/~spanel/segmentace/#_Toc125769336

[4] CANNY, John. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1986, PAMI-8, č. 6, s. 679-698. ISSN 0162-8828.

[5] The Canny Edge Detection and Its Improvement. *Artificial intelligence and computational intelligence: 4th International Conference, AICI 2012, Chengdu, China, October 26-28, 2012. Proceedings*. 1st ed. New York: Springer, 2012, s. 55-58. ISBN 978-3-642-33477-1.

[6] Image Transforms - Hough Transform. FISHER, Robert. *HIPR: hypermedia image processing reference* [online]. West Sussex, England: Wiley, c1996 [cit. 2013-02-22]. Dostupné z: http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm

[7] ADAMS, R. a L. BISCHOF. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 1994, roč. 16, č. 6, s. 641-647 [cit. 2013-02-23]. ISSN 01628828. DOI: 10.1109/34.295913. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=295913

[8] Color Image Segmentation Based on Blocks Clustering and Region Growing. BAO-LIANG LU, Liqing Zhang. *Neural information processing* [online]. Heidelberg [etc.]: SpringerLink, c2011, s. 459-466 [cit. 2013-02-23]. ISBN 978-3-642-24965-5.

[9] FARUQUZZAMAN, A.B.M. Object Segmentation Based on Split and Merge Algorithm. In: *2008 IEEE region 10 conference: Tencon 2008, Hyderabad, India, 19-21 November 2008*. Piscataway, NJ: IEEE, c2008, s. 1-4. ISBN 978-1-4244-2408-5.

[10] SHAO-YI, Chien. Predictive watershed: a fast watershed algorithm for video segmentation. *IEEE Transactions on Circuits and Systems for Video Technology* [online]. 2003, roč. 13, č. 5, s. 453-461 [cit. 2013-02-23]. ISSN 1051-8215. DOI: 10.1109/TCSVT.2003.811605. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1201120

[11] LINKA, Aleš, Petr VOLF a Miroslav KOŠEK. Segmentace prahováním. *E-Learning TUL* [online]. 2004 [cit. 2013-02-23]. Dostupné z: http://e-learning.tul.cz/cgi-bin/elearning/elearning.fcgi?ID_tema=67&ID_obsah=1206&stranka=publ_tema&akce=polozka_vstup

[12] *Pololu Maestro Servo Controller User's Guide*. 2001. Dostupné z: http://www.pololu.com/docs/pdf/0J40/maestro.pdf