VĚDECKÉ SPISY VYSOKÉHO UČENÍ TECHNICKÉHO V BRNĚ
Edice Habilitační a inaugurační spisy, sv. 198
ISSN 1213-418X

Jiří Šťastný

NETRADIČNÍ METODY A ALGORITMY PRO ROZPOZNÁVÁNÍ OBJEKTŮ TECHNOLOGICKÉ SCÉNY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta strojního inženýrství

RNDr. Ing. Jiří Šťastný, CSc.

Netradiční metody a algoritmy pro rozpoznávání objektů technologické scény

Nontraditional Methods and Algorithms for Object Recognition of Technological Scene

ZKRÁCENÁ VERZE HABILITAČNÍ PRÁCE

VUTIUM

BRNO 2006

KLÍČOVÁ SLOVA
rozpoznávání obrazců, detekce hran, algoritmus zpětného šíření, gramatika, Levenshteinova vzdálenost

KEYWORDS
Pattern Recognition, Edge Detection, Back-Propagation Algorithm, Grammar, Levenshtein distance

MÍSTO ULOŽENÍ PRÁCE
Oddělení pro vědu a výzkum Fakulty strojního inženýrství Vysokého učení technického v Brně

ISBN 80-214-3117-2
ISSN 1213-418X

OBSAH

PŘEDSTAVENÍ AUTORA

Jiří Šťastný se narodil 3. 9. 1953 v Brně. V roce 1978 absolvoval s vyznamenáním studium na Strojní fakultě VUT v Brně, obor Přístrojová, regulační a automatizační technika. V letech 1978-1979 působil jako odborný pracovník na Katedře strojního zařízení, mechanizace a automatizace stavebnictví FAST VUT v Brně. V letech 1979-1982 absolvoval vědeckou aspiranturu v oboru Stavba výrobních strojů a zařízení na FS VUT v Brně a v roce 1984 obhájil disertační práci na téma „Optimalizace parametrů konstrukce vlnového převodu s krokovým motorem pro pohony výrobních strojů a zařízení“. V roce 1982 nastoupil jako odborný asistent na Katedru přístrojové a automatizační techniky (dnešní Ústav automatizace a informatiky) na FS/FSI VUT v Brně, kde působí dodnes. V roce 1986 vystudoval při zaměstnání obor Matematická informatika na Přírodovědecké fakultě UJEP v Brně a v roce 1987 úspěšně složil ve stejném oboru státní rigorózní zkoušku (získal titul RNDr.). V letech 1989-1991 absolvoval odbornou stáž na odboru Výzkum a vývoj technologie VÚVL v Brně.

Jeho pedagogická praxe se datuje od roku 1980, kdy na Fakultě strojní VUT v Brně začal působit v předmětech Matematika I a Automatizace výrobních strojů. Od roku 1982 jako odborný asistent Katedry přístrojové a automatizační techniky začal vyučovat v předmětech Základy technické kybernetiky, Zařízení výpočetní techniky a Simulace a optimalizace dynamických systémů. Od té doby v rámci katedry a později Ústavu automatizace a informatiky prošel řadou dalších předmětů, např. Analýza systémů, Počítačová simulace, Počítačová podpora experimentů v životním prostředí, Databázové systémy aj. V současné době působí v předmětech Simulace systémů, Diplomový projekt, Zpracování informací a Informatika I. Je autorem nebo spoluautorem šesti skript v tištěné podobě a jedné studijní opory v elektronické podobě. Každoročně vede řadu diplomových prací, zaměřených na aplikace informačních technologií zejména v oblasti strojírenství. Doposud byl vedoucím více než 50ti obhájených diplomových prací. V současné době je školitelem tří doktorandů v oboru Konstrukční a procesní inženýrství.

Jeho vědeckovýzkumná a odborná činnost byla od samého počátku jeho působení na VUT zaměřena na oblast algoritmizace a programování, zejména strojírenských aplikací. Jedná se např. o realizovaný a na výstavě oceněný harmonický převod, vytvořený na základě původního algoritmu pro dimenzování harmonického kola, databázovou aplikaci v podobě informačního systému strojového parku pro odbor technologie VÚVL v Brně, projekt řešení datové komunikace PC se sítí pokladen OMRON ve spolupráci s firmou EPROS, nebo softwarový produkt „Teplárenský dispečink“ pro brněnskou teplárnu. Dlouhodobě spolupracuje na vývoji software pro řízení průmyslových robotů se specializací na rozpoznávání objektů. V posledních letech byl spoluřešitelem dvou grantů Grantové agentury ČR, projektu FRVŠ *Aktivní prvky vysokorychlostní sítě ATM a jejich využití v akademické síti, výuce a výzkumu* a výzkumného záměru *Netradiční metody studia komplexních a neurčitých systémů*. Výsledky své vědecké a odborné činnosti publikoval v 7 článcích ve vědeckých časopisech, ve 13 příspěvcích na světových vědeckých kongresech, ve 32 příspěvcích na národních, resp. evropských konferencích, ve 2 článcích v odborných knižnicích. Rovněž vypracoval 8 posudků na výzkumné projekty a publikace.

1 ÚVOD

Počítačové vidění (*computer vision*) patří v dnešní době k nejprogresivněji se rozvíjejícím oblastem informačních technologií. Klíčovým úkolem počítačového vidění je rozpoznání objektu. Metody rozpoznávání nalezly praktické uplatnění v celé řadě úloh, především diagnostické povahy.

Podle použitého popisu a způsobu vyhodnocování popisu objektů lze metody rozpoznávání rozdělit do dvou základních skupin. První skupina popisuje objekty souborem číselných charakteristik. Číselný vektor charakterizující objekt bývá označován jako *příznakový vektor* a metody, které jsou na něm založeny, jsou známé pod názvem *příznakové metody*. Příznakové metody nejsou vhodné tam, kde důležitým nositelem informace o objektech jsou strukturální vlastnosti objektů. Převedením úlohy do příznakového prostoru se totiž strukturální charakteristiky ztrácejí a je obtížné je získat zpět. V takovém případě je vhodnější popsat objekty pomocí elementárních popisných vlastností , tzv. primitiv a relací mezi nimi.

Metody, u nichž je každý objekt popsán relační strukturou, jsou známé pod názvem *strukturální (syntaktické) metody*. Omezením popisných relací na jedinou – *„následuje za“*, přejde struktura na posloupnost, neboli řetězec symbolů. Každý symbol odpovídá jednomu popisnému primitivu ze souboru primitiv, použitých k popisu. Tento soubor lze chápat jako abecedu formálního jazyka a řetězce jako slova formálního jazyka. Úloha rozpoznávání pak přechází na problém určit, zda slovo generované danou gramatikou odpovídá řetězci, popisujícímu daný objekt. Zatímco u příznakových metod rozpoznávání je využíván kvantitativní popis předmětů číselnými parametry (příznakovým vektorem), u strukturálních metod má vstupní popis kvalitativní charakter odrážející strukturu objektu.

Jeden ze způsobů, jak rozpoznat strukturu daného neznámého obrazu, spočívá v porovnání jeho strukturální reprezentace ve formě řetězce s reprezentacemi vzorových obrazů jednotlivých tříd. Takový způsob je nezbytný například v úlohách, kdy počet trénovacích vzorků je nedostatečný pro odvození gramatik, nebo když každý obraz může být považován za prototyp třídy obrazů. Tyto případy lze řešit pomocí algoritmů pro zjištění vzdálenosti mezi atributovými popisy obrazů.

Vlastní průběh zpracování a rozpoznávání digitalizovaného obrazu se obvykle dělí do několika základních kroků:

1. *Předzpracování*
2. *Segmentace obrazu*
3. *Popis objektů*
4. *Klasifikace (rozpoznávání) objektů*

Při posuzování algoritmů počítačového zpracování obrazu se berou v úvahu jejich časové a paměťové požadavky, které jsou klíčové zejména s ohledem na aplikační nasazení při řízení průmyslových robotů. Klíčovými body pro uvedenou aplikační oblast a tedy i pro tuto práci jsou výše uvedené body *3. (popis objektů)* a *4. (rozpoznávání objektů)*. Implementované netradiční algoritmy pro rozpoznávání objektů (neuronová síť, algoritmus pro zjištění vzdálenosti mezi atributovými popisy obrazů) jsou porovnávány s klasickou momentovou metodou.

Vzhledem k uvažovanému aplikačnímu nasazení u průmyslových robotů byly pro práci stanoveny následující úkoly:

1. navrhnout a implementovat původní simulační prostředí pro testování objektů technologických scén,
2. navrhnout a implementovat algoritmy pro netradiční metody rozpoznávání objektů technologické scény,

3. provést analýzu simulačních experimentů s ohledem na uvažované aplikační nasazení při řízení průmyslových robotů.

2 PŘEDZPRACOVÁNÍ OBRAZU

Metody předzpracování obrazu slouží ke zlepšení obrazu z hlediska dalšího zpracování. Cílem předzpracování je potlačit šum vzniklý při digitalizaci a přenosu obrazu, odstranit zkreslení dané vlastnostmi snímacího zařízení, případně potlačit nebo zvýraznit jiné rysy důležité z hlediska dalšího zpracování.

Mezi základní metody předzpracování obrazu je možné zahrnout:

a) *Převedení na stupně šedi (grey scale transformation)*. Při této operaci dochází k transformaci jedné hodnoty jasu pixelu na jinou bez ohledu na jeho pozici v obraze [7], [13], [34]. Hodnota 0 odpovídá černé barvě, hodnota 255 pak barvě bílé.

b) *Jas a kontrast (brightness correction)*. Jasové transformace slouží pro korekci jasové funkce [10], [13], [34], která může obsahovat zkreslení způsobené např. nerovnoměrným osvětlením scény, nestejnoměrnou citlivostí CCD prvku kamery, apod.

c) *Ekvalizace histogramu (histogram equalization)*. Jedná se o jednu z nejpoužívanějších metod pro vyrovnání jasových poměrů v obraze [7], [34]. Po úpravě je obraz kontrastnější a vystoupí detaily.

d) *Ostření obrazu (image sharpening)*. Cílem ostření obrazu [10], [34] je upravit obraz tak, aby v něm byly strmější hrany. Pro ostření obrazu se obvykle používají gradientní operátory [18].

e) *Filtrace (filtration)*. Filtrací [10], [34], [42] je označována skupina transformací, které převádějí hodnoty jasu vstupního obrazu na jiné jasové hodnoty s cílem zvýraznit nebo potlačit některé jeho vlastnosti. Často se provádí vyhlazování šumu v obraze.

3 SEGMENTACE OBRAZU

Proces segmentace rozčlení obraz na jednotlivé části, které mají úzkou souvislost s jednotlivými objekty [19], [34].

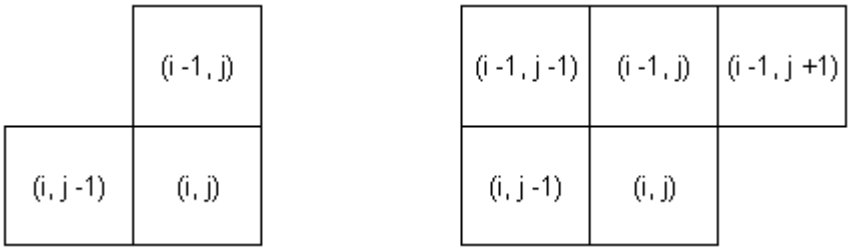
3.1 PRAHOVÁNÍ

Prahování [7], [34] je nejjednodušší segmentační postup. Vychází ze skutečnosti, že mnoho objektů nebo oblastí obrazu má konstantní odrazivost nebo pohltivost povrchu. Pak se může využít určená jasová konstanta *(práh)* k oddělení objektů od pozadí. Vzhledem k jednoduchosti výpočtu je prahování nejrychlejší segmentační metodou, kterou lze provádět v reálném čase.

Obr. 3.1 Prahování vhodným prahem, nízkým prahem a vysokým prahem

3.2 BARVENÍ

U této metody [10], [34] se obraz prochází po řádcích. Každému nenulovému obrazovému elementu $f(i, j)$ se přiřadí hodnota podle hodnoty jeho sousedních elementů, pokud sousední elementy existují (poloha sousedů je dána maskou).



Obr. 3.2 Masky pro barvení

3.3 DETEKCE HRAN

Segmentace na základě detekce hran vychází ze skutečnosti, že hranice objektů sestávají z hran, které jsou v obraze nalezeny aplikací některého z hranových operátorů (Laplaceův operátor, Sobelův operátor, operátor Prewittové aj.) [12], [19], [34]. Změna obrazové funkce může být popsána *gradientem.* Gradient je směr největšího růstu obrazové funkce (od černé po bílou). Hrany jsou kolmice na směr gradientu.

Dělení gradientních opererátorů :

a) Operátory aproximující derivace pomocí diferencí. Operátory invariantní vůči rotaci se realizují jedinou konvoluční maskou.

b) Operátory odhadující první derivaci používají několik masek. Směr gradientu se odhaduje hledáním masky, která odpovídá největší velikosti gradientu.

c) Operátory založené na hledání hran v místech, kde druhá derivace obrazové funkce prochází nulou *(zero crossing)*.

d) Operátory snažící se aproximovat obrazovou funkci jednoduchým parametrickým modelem, např. polynomem dvou proměnných.

3.4 VYPLŇOVÁNÍ OBJEKTŮ

Některé metody popisu objektů vyžadují znalost všech vnitřních bodů daného objektu. Z tohoto důvodu je potřeba objekt na kterém byla provedena segmentace pomocí detekce hran vyplnit a obarvit. Z důvodu časové náročnosti při vyplňování oblastí, je u metod, které vyžadují znalost vnitřních bodů objektu, výhodnější vycházet z metody barvení objektů [10], [34].

4 POPIS OBJEKTŮ

Klíčovou fází rozpoznávání obrazu je popis nalezených oblastí. Popsat tvar objektu není jednoduché. Je potřeba nalézt takový popis, který bude invariantní vůči posunu, rotaci a případně i velikosti.

4.1 MOMENTY OBJEKTU

Momentový popis oblastí [13], [34] interpretuje normalizovanou jasovou funkci obrazu jako hustotu pravděpodobnosti dvojrozměrné náhodné veličiny. Vlastnosti této veličiny lze vyjádřit

prostřednictvím statických charakteristik – *momentů*, které lze použít k popisu binárních i šedotónových oblastí.

Metoda je založena na výpočtu sedmi momentových příznaků objektu [13].

Obecný *2D* moment funkce $f(x,y)$ řádu $(p+q)$ není invariantní vůči změně měřítka, posunutí, ani natočení. Pro digitální obraz je definován:

$$m_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} i^p j^q f(i,j) \tag{4-1}$$

kde *i, j* jsou souřadnice bodů oblasti. Uvedený moment je závislý na velikosti, posunutí i natočení. Moment nezávislý na poloze a natočení je:

$$\mu_{pq} = \sum_x \sum_y (x - x_t)^p (y - y_t)^q f(x,y) \tag{4-2}$$

kde x_t, y_t jsou souřadnice těžiště:

$$x_t = \frac{m_{10}}{m_{00}}, \quad y_t = \frac{m_{01}}{m_{00}} \tag{4-3}$$

Obecné momenty m_{pq} jsou vypočítány z rovnice (4-1).

Nezávislost momentů na změně měřítka lze zajistit normovanými centrálními momenty:

$$\theta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^r}, \quad \gamma = \frac{p+q}{2} + 1 \tag{4-4}$$

Momenty invariantní vůči posunu, rotaci a změně měřítka se vypočítají z normovaných centrálních momentů θ_{pq} řádu *(p, q)* (4-4).

Pro výpočet momentů je nutné znát pozici všech pixelů uvnitř uzavřené hranice. Proto je třeba použít některou z metod pro vyplňování objektů [30], [31], nebo vycházet z metody barvení objektů [10], [34].

4.2 RAMENA OBJEKTU

V této metodě popisu odpovídají jednotlivé složky příznakového vektoru délkám ramen vektorů v polárním souřadnicovém systému s počátkem v těžišti právě prohledávaného objektu. Algoritmus vychází z obarvené scény [10], [34].

Algoritmus získání příznakového vektoru z ramen objektu:

1. Jestliže nebyly prohledány všechny objekty, pak vyber další objekt a pokračuj krokem 2, jinak konec.
2. Ve zvoleném úhlu prodlužuj rameno z těžiště tak dlouho, dokud není dosažena hrana objektu, zapiš jeho délku. Ve vytvořeném programu se uvažuje 70 ramen pro každý objekt (úhlový krok je $\varphi = 5°8'$).
3. Jestliže je celkové úhlové otočení ramene menší než 360°, otoč rameno o zvolený úhel a pokračuj krokem 2, jinak pokračuj krokem 4.
4. Vyhlaď vektor příznaků. Vyhlazení vektoru příznaků znamená nahrazení jednotlivých složek vektoru aritmetickým průměrem z *N* sousedních ramen. V programu je nastavena

hodnota 5 (aktuální rameno + 2 ramena z každé strany). Tím se zajistí vyhlazení obrysové křivky.

5. Zjisti velikost nejdelšího ramena a touto hodnotou vyděl všechny příznaky. Transformuj výsledné podíly do intervalu $\langle 0,\ 255 \rangle$. Tím se zajistí invariance vůči změně měřítka.
6. Pokračuj krokem 1.

Takto získaný vektor dostatečně dobře charakterizuje tvar objektu a umožňuje popis i složitějších tvarů. Jeho nevýhodou však je ignorování vnitřních hran.

4.3 GRAMATIKY

U syntaktického přístupu [15] , [27] je obraz reprezentován řetězcem, tvořeným sledem primitiv obrazu a vztahů mezi nimi.

Je-li objekt popsán hranicí, která je uzavřená, lze snadno nalézt primitiva. Těmito primitivy je pak definován tvar objektu. Takto zpracovaný obraz se dále používá v rozpoznávacích metodách.

U syntaktické analýzy obrazu je zkoumaný objekt rozdělen na konečný počet elementárních částí, představovaných primitivy. Primitivem může být např. přímka, úsečka, křivka, oblouk apod. Každému primitivu je přiřazen určitý symbol, který se nazývá terminál. Konečná množina terminálů bývá označována symbolem Σ, přičemž symbol Σ^* označuje množinu řetězců (slov) nad množinou Σ. Vztahy mezi primitivy jsou popsány nejčastěji jednorozměrnými relacemi. Příkladem jednorozměrné relace je relace uspořádání, která představuje zřetězení terminálů.

Řetězec terminálních symbolů (slovo) může reprezentovat daný obraz. Množina slov, popisujících obrazy jedné třídy tvoří jazyk této třídy. Jazyk, pokud je konečný, může být definován množinou jeho slov. Jednotlivá slova jsou generována tzv. přepisovacími pravidly gramatiky [15].

4.3.1 Generativní gramatika

Je definována [15] jako uspořádaná čtveřice $G = (N, \Sigma, P, S)$, kde :

N - konečná množina neterminálních symbolů

Σ - konečná množina terminálních symbolů

N a Σ - disjunktní konečné abecedy

P - konečná množina přepisovacích pravidel typu $\alpha \rightarrow \beta$

S - počáteční symbol gramatiky, $S \in N$

α, β - slova z abecedy $N \cup \Sigma$, α obsahuje alespoň jeden symbol z množiny N

Generativní gramatika $G = (N, \Sigma, P, S)$ určuje přepisovací systém $(N \cup \Sigma, P)$.

Jazyk generovaný gramatikou je definován jako množina řetězců (slov), které vyhovují podmínkám:

1. Každý řetězec je složen pouze z terminálních symbolů
2. Každý řetězec je odvozen z počátečního symbolu *S* použitím přepisovacích pravidel z množiny *P*.

Jazyk *L(G)* generovaný gramatikou *G* je definován zápisem:

$$L(G) = \{w;\ w \in \Sigma^* \text{ a } S \rightarrow^* w\} \qquad (4\text{-}5)$$

Je tedy tvořen všemi slovy v terminální abecedě, která lze odvodit z počátečního symbolu.

4.3.2 Typy gramatik

Některé jazyky lze generovat gramatikami [15], které mají přepisovací pravidla speciálního typu, tj. nevyužívají všech možností nabízených obecnou definicí gramatiky (viz 4.3.1).

Gramatika typu 1 – kontextová gramatika:

Je gramatika $G = (N, \Sigma, P, S)$, jejíž přepisovací pravidla mají tvar: $\alpha X \beta \rightarrow \alpha \gamma \beta$, kde: $\alpha, \beta \in (N \cup \Sigma)^*$, $X \in N$ a $\gamma \in (N \cup \Sigma)^+$ $\left(\text{tzn. } |\gamma| \geq 1\right)$. Jedinou výjimkou může být pravidlo $S \rightarrow e$, jehož výskyt znamená, že se *S* nesmí objevit na pravé straně žádného přepisovacího pravidla z množiny *P*.

Jazyk typu 1 *(kontextový jazyk)* je jazyk, generovaný kontextovou gramatikou.

Gramatika typu 2 – bezkontextová gramatika:

Je gramatika $G = (N, \Sigma, P, S)$, jejíž přepisovací pravidla mají tvar: $X \rightarrow \gamma$. kde: $X \in N$ a $\gamma \in (N \cup \Sigma)^*$.

Bezkontextová gramatika se nazývá *nevypouštějící,* jestliže neobsahuje žádné přepisovací pravidlo typu: $X \rightarrow e$, kde *e* označuje prázdný řetězec.

Ke každé bezkontextové gramatice *G* existuje *nevypouštějící bezkontextová gramatika* G_1 taková, že platí: $L(G_1) = L(G) - \{e\}$

Jazyk typu 2 *(bezkontextový jazyk)* je jazyk, generovaný bezkontextovou gramatikou.

Gramatika typu 3 – regulární gramatika:

Je gramatika $G = (N, \Sigma, P, S)$, jejíž přepisovací pravidla mají tvar: $X \rightarrow wY$ nebo $X \rightarrow w$, kde: $X, Y \in N$, $w \in \Sigma^*$ (*w* je tedy řetězec terminálů).

Jazyk typu 2 *(regulární jazyk)* je jazyk, generovaný regulární gramatikou. Tyto jazyky jsou rozpoznatelné konečnými automaty. Slovo generované regulární gramatikou *G* je rozpoznatelné konečným automatem *A,* jestliže platí: $L(G) = L(A)$.

Pomocí regulárních gramatik lze řešit veškerou problematiku popisu objektů z dané aplikační oblasti.

4.3.3 Návrh primitiv

Primitiva [20], [27]. jsou základním stavebním prvkem obrazu u strukturálních metod rozpoznávání. Při jejich návrhu je vhodné zvážit možnost jejich snadného rozpoznání. Pro obrazy, které jsou charakterizovány hranicí, jsou vhodnými primitivy části čar. Např. úsečka může být charakterizována svým počátkem a koncem, délkou, případně úhlem. Podobně mohou být charakterizovány i části křivek. Návrh primitiv závisí na řešené aplikaci. Obecně platí:

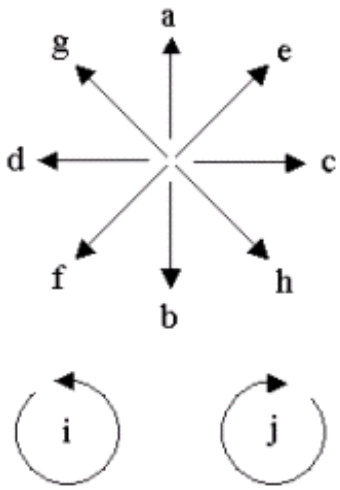
a) Primitiva musí být snadno rozpoznatelná.

b) Primitiva musí poskytovat kompaktní a postačující popis obrazů pomocí specifikovaných vztahů.

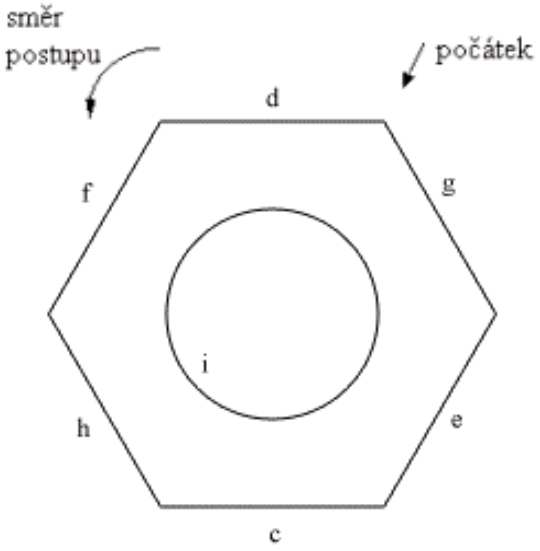
V případě použití složitějších primitiv dostaneme jednodušší strukturální popis objektu, což sice představuje použití jednodušší gramatiky pro popis objektu, ale důsledkem je zvýšení složitosti při

hledání takových primitiv v obraze. Naopak jednodušší primitiva vedou sice ke složitější gramatice, jejich výhodou však je jednodušší identifikace v obraze.

Po návrhu primitiv je dalším důležitým krokem sestavení přepisovacích pravidel pro gramatiku na základě potřebných znalostí i zkušeností. Pro danou aplikaci byla navržena primitiva daná směry řetězcových kódů (viz obr. 4.1). Ukázka popisu objektu danými primitivy je na obr. 4.2.



Obr. 4.1 Směry řetězcových kódů



Obr. 4.2 Popis objektu s vnitřní hranou

5 ROZPOZNÁVÁNÍ OBJEKTŮ

Rozpoznávání objektů (klasifikace) [33], [34], [35] spočívá v zařazování objektů do tříd. *Třída* je podmnožina prvků, jejichž atributy mají z hlediska klasifikace společné rysy. Objekt v počítačovém vidění představuje nejčastěji části segmentovaného obrazu.

Samotnou činnost klasifikace vykonává *klasifikátor*. Klasifikátor nerozhoduje o třídě objektu podle objektu skutečného, nýbrž podle jeho obrazu.

5.1 ROZPOZNÁVÁNÍ PŘÍZNAKOVĚ POPSANÝCH OBJEKTŮ

Příznakový popis objektu [33], [34] se vyznačuje číselným charakterem elementárních popisů. Elementární popisy se nazývají *příznaky*. Příznaky lze získat pomocí metod popsaných v kap. 4.1 a 4.2.

5.1.1 Rozpoznávání na principu minimální vzdálenosti

Mezi základní a nejčastější metody patří *Hammingova vzdálenost* [33], [34], která je jednoduchá a snadno se realizuje. Hammingova metrika hledá rozdíly mezi jednotlivými vektory, tj. rozdíly mezi jednotlivými elementy a celková vzdálenost je pak dána součtem hodnot těchto rozdílů:

$$H = \sum_{i=1}^{N} |a_i - b_i| \tag{5-1}$$

Ze vztahu je patrné, že vzdálenost je dána součtem všech rozdílných bodů dvou obrazců.

Ve vytvořeném testovacím prostředí byla použita jedna z nejznámějších metrik - *Euklidova vzdálenost (E)* [33], [34], která předpokládá kartézský souřadný systém se dvěma vektory *A* a *B*. Pak obecně pro prostor dimenze *N* platí:

$$E = \sqrt{\sum_{i=1}^{N} \left(A(i) - B(i) \right)^2} \tag{5-2}$$

Prakticky se výpočet minimální hodnoty Euklidovy vzdálenosti v programu provádí výpočtem součtu absolutních hodnot rozdílů odpovídajících složek příznakových vektorů etalonu a neznámého objektu. Výsledný součet *n* sčítanců je vyčíslen jako EV_1. Pak se provede rotace příznakového vektoru neznámého objektu tak, že jeho první složka se posune na místo druhé, druhá na místo třetí atd., až *n* – tá složka se přesune na místo první. Poté se opět provede součet rozdílů a označí se EV_2. Analogicky se vypočítají hodnoty EV_3, …, EV_n. Minimální hodnota ze všech Euklidových vzdáleností zjišťovaná přes všechny etalony určuje zařazení neznámého objektu do správné třídy.

5.2 NEURONOVÉ SÍTĚ

Hlavní rozdíl od klasických výpočetních postupů spočívá v tom, že u neuronových sítí [24], [33] nemusíme znát algoritmus řešení daného problému. Postačuje znalost určitého počtu příkladů a jejich řešení.

Neuronové sítě lze využívat samostatně i ve spojení se standardními prostředky pro zpracování informací (databázové systémy, expertní systémy aj.).

Umělé neuronové sítě se snaží napodobit chování a funkci biologických neuronů. Za umělou neuronovou síť je obecně považována taková struktura pro distribuované paralelní zpracování dat, která se skládá z určitého, vysokého počtu vzájemně propojených výkonných prvků *(neuronů)*. Každý z nich přitom může současně přijímat libovolný konečný počet shodných informací o stavu svého jediného výstupu. Každý neuron transformuje vstupní hodnoty alespoň dvěma funkcemi (výpočetními procedurami). První je *aktivační funkce*, která generuje hodnoty vstupů, a druhá *přenosová funkce* (nejčastěji skoková, sigmoidální nebo Gaussova), která převádí hodnotu výstupu aktivační funkce do definovaného oboru výstupních hodnot (nejčastěji do intervalu $\langle 0,1 \rangle$).

Neuronová síť pracuje v zásadě ve dvou fázích – adaptivní a aktivní. V adaptivní fázi se síť učí, v aktivní vykonává naučenou činnost. Při učení neuronové sítě dochází ke změnám, síť se adaptuje

na řešení daného problému. Učení neuronové sítě je obvykle realizováno nastavováním vah vstupů jednotlivých neuronů. Rozlišují se dva typy učení - *s učitelem* a *bez učitele*.

Při učení s učitelem existuje nějaké vnější kritérium, které určuje, který výstup je správný. V praxi se to řeší sadou testovacích příkladů (*trénovací množina*), ke kterým známe řešení. Tyto příklady postupně předkládáme na vstup neuronové sítě a srovnáváme skutečnou odezvu sítě s požadovanou. Na základě odlišnosti skutečného výstupu od očekávaného pak pomocí zpětné vazby upravujeme váhy v neuronové síti. Metodika úpravy vah je dána učícím algoritmem. Je dokázáno, že po provedení velkého počtu učících se kroků se síť naučí poskytovat stabilní výstup jako reakci na naučené vstupy.

Pro danou aplikační oblast byly testovány různé typy neuronových sítí [36], přičemž nejlepší výsledky dávala síť MLP (Multi Layer Perceptron).

5.2.1 Vrstvená perceptronová síť (Multi Layer Perceptron)

Vrstvená perceptronová síť je acyklická dopředná síť. Neurony lze disjunktně rozdělit do vrstev tak, že výstupy každého neuronu jedné vrstvy jsou napojeny na vstupy každého neuronu vrstvy následující. Neexistují žádné vazby mezi neurony nesousedních vrstev ani mezi neurony stejné vrstvy. Každý neuron má právě tolik vstupů, kolik je neuronů v nižší vrstvě. Vstupní vrstva slouží pouze k distribuci vstupních hodnot do první skryté vrstvy. Síť s jednou skrytou vrstvou a jednou vrstvou výstupní se označuje jako síť *dvouvrstvá,* síť se dvěma skrytými vrstvami jako *třívrstvá,* atd. [24].

Jako učící algoritmus pro neuronovou síť MLP (Multi Layer Perceptron) byl testován algoritmus *Back-propagation* a g*enetický algoritmus* [39], přičemž lepší výsledky pro danou aplikaci dával algoritmus Back-propagation.

Algoritmus Back-propagation je iterační proces, ve kterém se síť dostává z počátečního nenaučeného stavu do stavu úplného naučení. Princip algoritmu spočívá v testování, zda neuronová síť odpovídá na vstupní vektor přesně podle trénovací množiny. V případě, že síť nereaguje podle požadavků, je nutné měnit váhové koeficienty tak dlouho, dokud nezačne reagovat správně.

Algoritmus Back-propagation:

```
náhodná_inicializace_vah;
repeat
    repeat
        vyber_vzor_z_trénovací_množiny;
        přilož_vybraný_vzor_na_vstup_sítě;
        vypočti_výstupy_sítě;
        porovnej_výstupy_s_požadovanými_hodnotami;
        modifikuj_váhy;
    until vybrány_všechny_vzory_z_trénovací_množiny;
until globální_chyba < kritérium;
```

Algoritmus Back-propagation je založen na minimalizaci energie neuronové sítě. Energie je mírou naučenosti, tedy odchylky mezi skutečnými a požadovanými hodnotami výstupů neuronové

sítě pro danou trénovací množinu. Během učení neuronové sítě dochází k poklesu energetické funkce. *Energie sítě,* která je součtem druhých mocnin odchylek, se vypočte dle vztahu:

$$E = \frac{1}{2}\sum_{i=1}^{n}\left(u_i - d_i\right)^2 \qquad (5\text{-}3)$$

kde n je počet výstupů sítě, u_i je skutečný i-tý výstup a d_i je požadovaný i-tý výstup.

Pro dobře naučenou síť konverguje chyba po konečném počtu učících cyklů k nule. Bohužel existují případy, kdy se nepodaří síť naučit s dostatečně malou chybou. V takovém případě dosáhne energie určité hodnoty a dál už neklesá. Tomuto stavu se říká *uvíznutí v lokálním minimu.* Každý další krok potom vede ke vzestupu energie.

Problém uvíznutí v lokálním minimu lze řešit:

a) Vhodnou volbou parametru η *(koeficient učení)* a μ *(koeficient vlivu změny vah)*. Volbou větší hodnoty parametru η bude krok tak velký, že malá lokální minima budou přeskočena. Se zvětšováním hodnoty parametru η však vzrůstá riziko oscilace. Účinnější je koeficient μ. Ten přidává do postupu váhovým prostorem setrvačnost. Při dosažení lokálního minima zajistí tato setrvačnost udržení směru, v němž bylo tohoto minima dosaženo. Tento směr se zachovává i přes nárůst energie a lokální minimum bude překonáno.

b) Vhodnou strategií výběru vzorů z trénovací množiny. Nejméně odolnou strategií z hlediska uvíznutí v lokálním minimu je sekvenční výběr vzorů. Naopak účinnější metodou se jeví náhodný výběr.

c) Vhodným počátečním nastavením vah. Ty musí být dostatečně malé, náhodně generované, z doporučeného [23], [33] intervalu $\langle -0.3, 0.3 \rangle$.

5.3 ROZPOZNÁVÁNÍ SYNTAKTICKY POPSANÝCH OBJEKTŮ

Zatímco v příznakových metodách rozpoznávání je využíván kvantitativní popis objektů číselnými parametry – příznakovým vektorem, v syntaktických metodách [20], [27] má vstupní popis kvalitativní charakter odrážející strukturu objektu. Elementární vlastnosti syntakticky popsaných objektů - *primitiva* - jsou části hranice určitého tvaru, příp. grafový nebo relační popis oblastí, kdy primitiva představují podoblasti určitého tvaru.

Úkolem syntaktického rozpoznávání obrazu je určit, zda analyzovaný obraz odpovídá obrazům dané gramatiky, tj. zda gramatika může tento obraz generovat. Obraz je reprezentován řetězcem jazyka, který je generován danou gramatikou.

Nejjednodušším způsobem rozpoznávání je „porovnávání se vzorem“. Řetězec reprezentující obraz je porovnáván s prvky množiny řetězců, představující jednotlivé vzorové obrazy. Při porovnávání je nutná buď úplná shoda se vzorem nebo jen částečná shoda, ale na základě určitého přizpůsobovacího kritéria. Tento způsob je jednoduchý a rychlý. Je-li však třeba pro rozpoznání úplný popis obrazu, je nezbytná *syntaktická analýza*.

Při návrhu *syntaktického analyzátoru* je vhodné předpokládat náhodné vlivy, např. deformace obrazu.

5.3.1 Metody pro zjištění vzdálenosti mezi atributovými popisy obrazů

Jeden ze způsobů, jak rozpoznat strukturu daného neznámého obrazu, spočívá v porovnání jeho strukturální reprezentace ve formě řetězce, stromu, či relačního grafu s reprezentacemi vzorových obrazů jednotlivých tříd. Takový způsob je nezbytný například v úlohách, kdy počet trénovacích

vzorků je nedostatečný pro odvození gramatik, nebo když každý obraz může být považován za prototyp třídy obrazů.

Pro stanovení vzdálenosti mezi dvěma řetězci lze použít metody pro zjištění vzdálenosti mezi atributovými popisy obrazů.

Analýza metod pro zjištění vzdálenosti mezi atributovými popisy obrazů

Z dostupných metod pro zjištění vzdálenosti mezi atributovými popisy obrazů byly analyzovány metody: Hammingova vzdálenost $H_d(s,t)$ [5], [16], Levenshteinova vzdálenost $L_d(s, t)$ [5], [16], [21], Daemerauova vzdálenost $D_d(s, t)$ [16], [21], Jaccardova vzdálenost J_d [5], [16], Minkowského vzdálenost $M_d(s,t,power)$ [16], [21] a metoda Needleman-Wunsch [5], [16], [21].

Nevhodné pro uvažovanou aplikační oblast jsou metody Hammingova a Jaccardova, protože nejsou schopny porovnávat řetězce nestejných délek, což pro zjištění vzdálenosti mezi atributovými popisy obrazů je nepostradatelná podmínka (rozlišené popisné řetězce nalezených vzorků se mohou značně lišit od řetězcových popisů etalonů vlivem nejrůznějších poruch obrazu, které vzniknou v procesu předzpracování obrazu).

U Minkowského metody je použití pro aplikaci na zjištění vzdálenosti mezi atributovými popisy obrazů velmi problematické, neboť dynamicky počítané ohodnocení ceny za operaci podle vzdálenosti porovnávaných znaků (vzdálenost v ASCII tabulce) může zapříčinit značné výkyvy výsledné vypočítané hodnoty vzdáleností.

Zbývající tři metody Levenshtein, Damerau a Needleman-Wunsch jsou použitelné pro zjištění vzdálenosti mezi atributovými popisy obrazů. U metody Damerau však přidaná operace výměny dvou sousedních znaků pro uvažované aplikace zbytečně komplikuje implementaci algoritmu. U Needleman-Wunsch metody je možnost ohodnocení různých operací cenami obtížně využitelná, nezpůsobuje však velkou komplikaci algoritmu.

Na základě analýzy uvedených metod byly jako nevhodnější pro zjištění vzdálenosti mezi atributovými popisy obrazů vybrány metody Levenshteinova a Needleman-Wunsch.

Žádný z uvedených algoritmů není invariantní vůči natočení, proto každému musíme předkládat vždy jeden z popisných řetězců tolikrát, kolik znaků představuje jeho délka, přičemž se v každém kroku provede rotace řetězce o jeden znak. Pak se vybere nejkratší zjištěná řetězcová vzdálenost. Tím se časová složitost každého z algoritmů zvýší v závislosti na délce rotujícího řetězce.

Levenshteinova vzdálenost (Levenshtein distance)

Levenshteinova vzdálenost $L_d(s, t)$ [5], [16], [21] je definována jako míra podobnosti mezi dvěma řetězci *s* a *t*. Levenshteinova vzdálenost $L_d(s, t)$ je počet vložení, smazání a substitucí znaků potřebných k transformaci řetězce *t* na řetězec *s*.

Příklady:

Příklad 1: Řetězce jsou identické

s: d b d f b c a j b c a

t: d b d f b c a j b c a

Levenshteinova vzdálenost $L_d(s, t) = 0$

Příklad 2: Vložení (viz obr. 5.1)

s: d b d f b c a j b c a

t: d b d f b c a j - c a

vložení znaku ‘b’, Levenshteinova vzdálenost $L_d(s, t) = 1$

Příklad 3: Smazání

s: d b d f b c a j b c a

t: d b d f b c a j a b c a

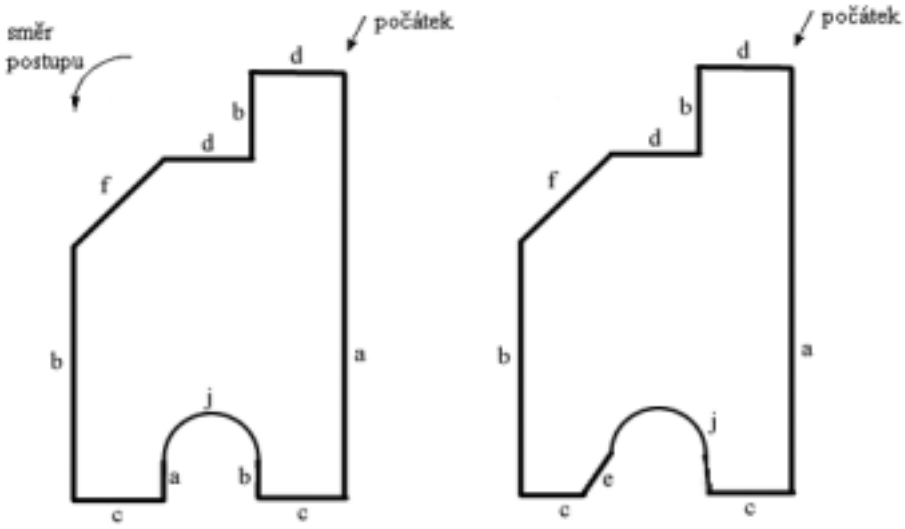
smazání znaku ‘a’, Levenshteinova vzdálenost $L_d(s, t) = 1$

Příklad 4: Substituce (viz obr. 5.1)

s: d b d f b c a j b c a

t: d b d f b c e j b c a

substituce ‘a’ -> ‘e’, Levenshteinova vzdálenost $L_d(s, t) = 1$



Obr. 5.1 Popis etalonu a objektu

Implementovaný algoritmus:

Krok 1

Pokud je jeden z porovnávaných řetězců prázdný, algoritmus končí s výsledkem maximální délky – délka nenulového řetězce.

Krok 2

Naplnění nultého řádku matice vzdáleností *d* inicializačními hodnotami.

For k<-0 to n do

d[k] <- k

Krok 3

Naplnění nultého sloupce matice vzdáleností *d* inicializačními hodnotami.

For k<-0 to m do

*d[k*n] <- k*

Krok 4

Doplnění matice vzdáleností *d* dopočítáním hodnot ve dvou vnořených cyklech.

For i <- 1 to n do

For j <- 1 to m do

Krok 5

Nastavení ceny (*cost*) podle rovnosti znaků z konkrétní pozice v každém z řetězců. Pokud se znak řetězce *s* na pozici *[i-1]* rovná znaku z řetězce *t* na pozici *[j-1]*, potom se cena za operaci (*cost*) nastaví na 0, v opačném případě (tedy pokud se příslušné znaky nerovnají) se nastaví na hodnotu 1.

If s(i-1) = t(j-1)

then cost <- 0

else cost <- 1

Krok 6

Výpočet tří hodnot - vzdáleností, ze kterých se v *kroku 7* vybírá minimum. Jedná se o vzdálenosti potřebné k provedení operací vložení, smazání nebo substituce znaku.

*a <- d[(j-1) * n + i] + 1*

*b <- d[j * n + i – 1] + 1*

*c <- d[(j-1) * n + i - 1] + cost*

Krok 7

Vybere se nejmenší hodnota ze tří hodnot, vypočtených v *kroku 6* a zapíše se na příslušnou pozici v matici vzdáleností *d*.

*d[j * n + i] <- (min(a,(min(b,c))))*

Krok 8

Konečným výsledkem je hodnota, vypočtená na poslední pozici v matici vzdáleností *d*.

*distance <- d[n * m - 1];*

Levenshteinova metoda poskytuje rozšířenou reprezentaci vzdálenosti mezi dvěma porovnávanými řetězci.

Needleman-Wunsch vzdálenost (Needleman-Wunsch distance)

Metoda Needleman-Wunsch [5], [16], [21] vypočítává vzdálenost mezi řetězci použitím matic cen za provedenou operaci. Do výsledku se promítnou ceny, které mohou být pro každou operaci různé hodnoty. Povolené operace vložení, smazání a substituce mohou mít tedy každá různou cenu. Pro každou operaci existuje jedna matice cen. Uvnitř matice cen můžeme definovat pro každou buňku zvláštní ohodnocení. Pro testovací účely však každá z matic cen bude naplněna stejnou hodnotou.

Metoda Needleman-Wunsch poskytuje podobnou reprezentaci vzdálenosti mezi dvěma porovnávanými řetězci jako Levenshteinova vzdálenost. Tuto metodu opět můžeme použít i pro nestejně dlouhé řetězce. Hlavní rozdíl ve srovnání s Levenshteinovou metodou je v možnosti ohodnocení různých operací cenami. Tento rozdíl však neposkytuje pro uvažované aplikace žádnou výraznou výhodu. Vzhledem k tomu, že časová náročnost této metody je přibližně stejná

jako u Levenshteinovy metody, byla metoda Needleman-Wunsch rovněž implementována ve vytvořeném simulačním prostředí pro testovací účely.

Algoritmy pro rozpoznávání obrazu pomocí vyhodnocování vzdáleností mezi atributově popsanými objekty, použité ve vytvořeném simulačním prostředí, probíhají ve dvou hlavních fázích:

1. Výpočet vzdáleností
2. Vyhodnocení vzdáleností

Výpočet vzdáleností:

V této fázi se porovnávají rozpoznané objekty s etalony. Výsledkem této fáze jsou všechny zjištěné vzdálenosti, které existují mezi všemi rozpoznanými objekty (včetně všech jejich rotací) a všemi etalony. Všechny vypočítané vzdálenosti nese každý objekt s sebou i s informací, jaká vzdálenost odpovídá danému etalonu (ve druhé fázi se provádí vyhodnocení na základě právě těchto vypočítaných vzdáleností).

Implementovaný algoritmus:

1. Jestliže nejsou analyzovány všechny objekty, načti další objekt a pokračuj krokem 2, jinak pokračuj krokem 6.
2. Jestliže nejsou analyzovány všechny etalony, načti další etalon a pokračuj krokem 3, jinak pokračuj krokem 1.
3. Jestliže nejsou vyčerpány všechny rotace řetězcové reprezentace objektu, rotuj objekt a pokračuj krokem 4, jinak pokračuj krokem 2.
4. Vypočítej řetězcovou vzdálenost mezi rotovanou řetězcovou reprezentací objektu a řetězcovou reprezentací etalonu podle zadaného algoritmu pro zjištění vzdálenosti mezi dvěma řetězci a pokračuj krokem 5.
5. Zapiš zjištěnou vzdálenost a pokračuj krokem 1.
6. Proveď druhou fázi – Vyhodnocení vzdáleností.

Vyhodnocení vzdáleností:

V této fázi se vyhodnotí „míra příslušnosti“ každého etalonu ke každému objektu podle vypočítaných vzdáleností z předcházející fáze. „Míra příslušnosti” je dána uživatelsky nastavitelnými parametry *hloubka* a *percentil* (viz dále).

Popis použitých nastavitelných parametrů:

Hloubka:

a) Jedná se o index hloubky minimálních vzdáleností (použitý v analýze ve druhé fázi).
b) Pro každý objekt je vypočítáno pole vzdáleností, kde jednotlivé vzdálenosti odpovídají zjištěné řetězcové vzdálenosti objektu od každého známého etalonu (objekt přitom s sebou nese informaci, jaká vzdálenost odpovídá danému etalonu).
c) Toto pole vzdáleností je uspořádáno vzestupně, tzn. první prvek pole odpovídá nejmenší zjištěné vzdálenosti objektu od určitého etalonu.
d) Nastavitelný parametr *Hloubka* je index pole vzdáleností a určuje potom hodnotu, která ovlivní rozhodování o tom, zda objekt vyhovuje etalonu či nikoliv (algoritmus probíhá ve druhé fázi analýzy).

Percentil:

a) Jedná se o procentuální ohodnocení „příslušnosti" objektu k etalonu.

b) Jeho hodnota je vypočítána jako poměr řetězcové vzdálenosti objektu od etalonu (*vzdálenost_objektu_od_etalonu* je vzdálenost vypočítaná zvoleným algoritmem pro zjištění vzdálenosti mezi dvěma řetězci) vydělená počtem znaků popisu reprezentující objekt, vyjádřená v procentech.

c) Je to tedy procentuální vyjádření počtu správných znaků řetězcového popisu objektu k řetězcovému popisu etalonu:

$$Percentil\_vypoč = 100 - \left(\frac{vzdálenost\_objektu\_od\_etalonu}{délka\_řetězce\_objektu} * 100 \right)$$

d) Vzorec pro *Percenti_vypoč* byl odvozen experimentálně.

e) Nastavitelný parametr *Percentil* ovlivňuje rozhodování o tom, zda objekt vyhovuje etalonu či nikoliv (algoritmus probíhá ve druhé fázi analýzy).

5.3.2 Metody syntaktické analýzy

Existují dvě základní metody [15] syntaktické analýzy. Jedná se o analýzu shora a analýzu zdola. Při syntaktické analýze je zpočátku znám pouze řetězec a počáteční symbol gramatiky, mezi nimiž se generuje invertovaná stromová struktura.

Analýza shora: Provádí se konstrukce stromu od jeho vrcholu (od počátečního symbolu) ke zkoumanému řetězci.

Analýza zdola: Provádí se konstrukce stromu od řetězce k počátečnímu symbolu.

Implementovaný algoritmus pro syntaktickou analýzu *(syntaktický analyzátor je navržen pro analýzu zdola)**:***

1. Jestliže nejsou analyzovány všechny řetězce, načti nový řetězec a pokračuj krokem 2, jinak pokračuj krokem 7.
2. Proveď analýzu zdola pro vybranou třídu.
3. Jestliže řetězec patří do jazyka gramatiky vybrané třídy, pokračuj krokem 6.
4. Jestliže počet rotování řetězce je menší než délka řetězce, rotuj řetězec a pokračuj krokem 2, jinak pokračuj krokem 5. Rotování řetězce znamená přesunutí terminálního symbolu z poslední pozice na první.
5. Jestliže počet otočení předmětu je menší než 360 / úhlový krok, otoč předmět o zadaný úhlový krok a pokračuj krokem 2. Rotování předmětu znamená pootočení předmětu o zadaný úhel a tím získání jiného řetězce.
6. zapiš výsledek a pokračuj krokem 1.
7. vypiš zprávu o rozpoznání objektu.

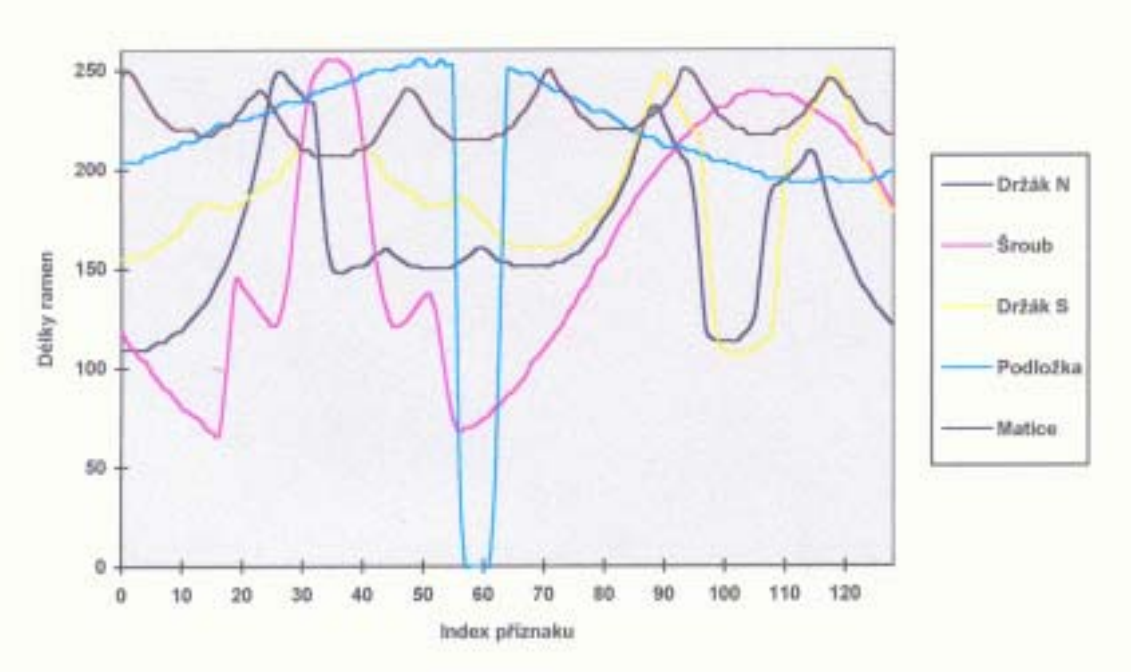
6 ANALÝZA SIMULAČNÍCH EXPERIMENTŮ

Ve vytvořeném simulačním prostředí byly testovány algoritmy pro rozpoznávání objektů na základě příznakového a strukturálního popisu. Záměrem bylo testovat takové objekty (obr. 6.1), které se podobají dvourozměrným obrazům reálných předmětů. Rovněž záměrně byly zvoleny i předměty tvarově blízké (držák N a držák S, částečně i matice s podložkou). Ukázky simulačních experimentů pro simulovanou technologickou scénu *Etalony.bmp* (obr. 6.2) jsou pro jednotlivé metody uvedeny v tabulkách.

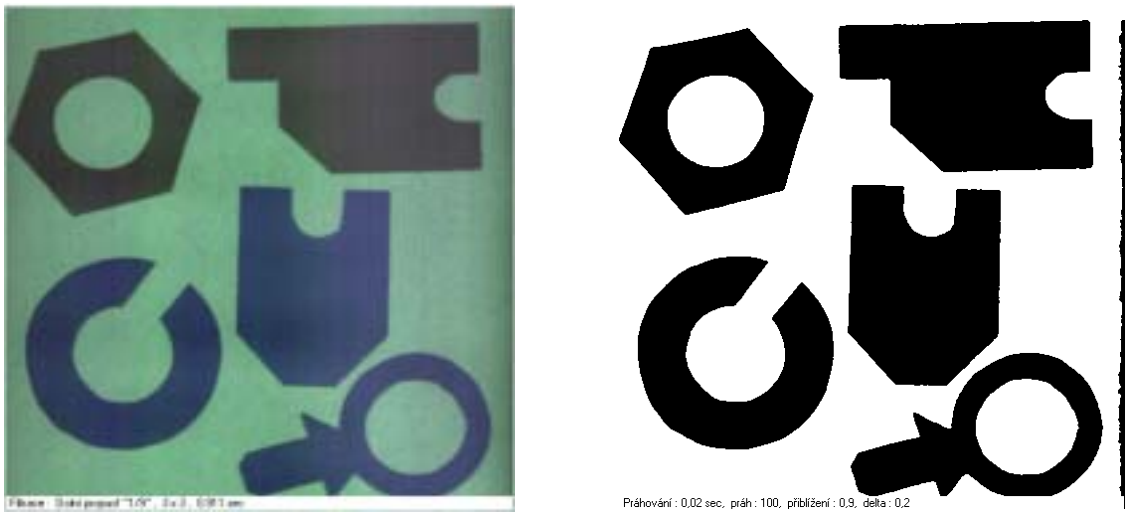
Obr. 6.1 Objekty pro testování

Obr. 6.2 Scéna *Etalony.bmp*

Grafické znázornění vektorů příznaků jednotlivých objektů vzorové scény (obr. 6.2) v příznakovém prostoru je na obr. 6.3. Příklad testované scény a jejího zpracování prahováním v experimentálním simulačním prostředí je znázorněn na obr. 6.4.



Obr. 6.3 Znázornění příznakových vektorů



Obr. 6.4 Testovaná scéna a prahování

6.1 HRANOVÁ DETEKCE OBJEKTŮ

Metody pro detekci hran byly srovnávány podle několika kritérií:

a) *Hladká hrana*: Udává čistotu hrany, zda neobsahuje zbytkový šum, zda šířka hrany má všude stejný počet pixelů, atd.

b) *Zdvojení hran*: Posuzuje se zdvojení hran, nebo dvojité zobrazení objektu s posunutím.

c) *Šum*: Posuzuje čistotu scény

d) *Tloušťka čar*: Tato položka zahrnuje šířku čáry po celém obvodu hrany. Posuzována byla změna šířky čáry v jednotlivých bodech i průměrná šířka čáry. Pro šířku čáry byl zvolen požadavek 2 pixely.

e) *Zachování tvaru*: Udává tvarovou modifikaci předmětu po detekci hran.

f) *Vymazání hran*: Sleduje chybné vymazávání celých úseků hran.

g) Poslední položka informuje o *nastavení vstupní hodnoty.*

Pro hodnocení metod podle navržených kriterií byl zvolen grafický přístup:

✓✓✓	Nejlepší výsledek. Výsledek splňuje nejlépe zadaná kritéria
✓✓	Téměř ideální výsledek. Plně postačující pro další zpracování
✓	Postačující minimum. Při tomto stavu už dochází k chybám při dalším zpracování, avšak scéna ještě není zcela nepoužitelná
✗	Od tohoto bodu již metoda nevyhovuje. Scéna již není použitelná pro další zpracování, nebo by další úpravy takového obrazu byly zbytečně náročné
✗✗	Další nevyhovující bod. Udává jen míru zhoršení obrazu oproti předchozí značce
✗✗✗	Nejhorší výsledek v určené kategorii. Tento už natolik ovliňuje scénu, že se scéna stává naprosto nepoužitelnou

Tab. 1 obsahuje čtyři metody pro detekci hran. Z tabulky vyplývá, že nejvhodnější metodou pro předem neupravenou scénu z těchto čtyř metod je *operátor Prewittové*, popř. *Gradientní operátor*. *Operátor Prewittové* vykazoval lepší výsledky než *Gradientní operátor* a naopak *Gradientní operátor* detekoval hrany desetkrát rychleji. Na rozdíl od *Gradientního operátoru* neobsahoval obraz po detekci hran *operátorem Prewittové* žádný šum, popř. obsahoval šum zanedbatelný.

	Operátor Prewittové	**Laplaceův operátor**	**Gradientní operátor**	**Průchod nulou**
čas (s)	0,25	1,125	0,025	0,05
hladká hrana	✓✓	✓	✓✓	✓
zdvojení hran	✓✓✓	✓	✓✓✓	✓✓✓
šum	✓✓✓	✗	✓	✗
tloušťka čar	✓✓	✗	✓✓	✓
zachování tvaru	✓✓✓	✓✓✓	✓✓✓	✓✓✓
vymazání hran	✓✓✓	✓✓✓	✓✓✓	✓✓✓
práh	165	100	31	193

Tab. 1 Metody pro detekci hran v obraze

V případě, že se použije před detekcí hran metoda *prahování*, pak generují všechny uvedené metody (*operátor Prewittové, Laplaceův operátor, operátor průchodu nulou*) naprosto stejné výsledky a mohou se tedy porovnávat podle rychlosti detekce hran. *Gradientní operátor* se liší generováním tenších, dva pixely širokých hran, čímž se dostává na první místo v rychlosti i v kvalitě výsledných detekovaných hran.

6.2 UKÁZKA VÝSLEDKŮ TESTOVÁNÍ PŘÍZNAKOVĚ POPSANÝCH OBJEKTŮ

Jak je vidět z tab. 2, rozpoznávání objektů *momenty* přineslo velmi dobré výsledky. Již při naučení na jednom vzoru etalonů tato metoda klasifikovala bezchybně většinu objektů. Pro výpočet momentů je však nutné znát pozici všech pixelů uvnitř uzavřené hranice, což výsledný čas zvyšuje.

Momenty						
scéna	čas (s)	klasifikace			etalony	výsledný soubor
		správně	chybně	neidentif.		
1	0,491	5 - etalony 1	0	0	1	ScénaMO 1a.bmp
2	0,47	5 - etalony 1	0	0	1	ScénaMO 2a.bmp
3	0,471	4 - etalony 1	0	1	1	ScénaMO 3a.bmp
4	0,501	3 - etalony 1	2 - etalony 1	0	1	ScénaMO 4a.bmp
5	0,22	1 - etalony 1	6 - etalony 1	1	1	ScénaMO 5a.bmp

Tab. 2 Rozpoznávání pomocí momentů

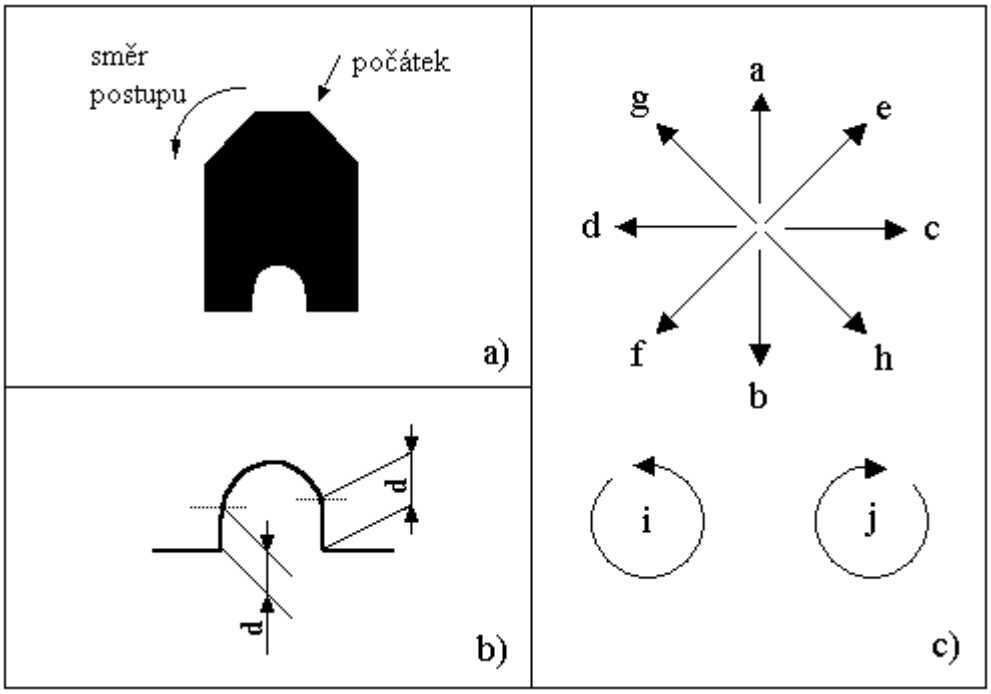
Tab. 3, je přehledem výsledků klasifikace pomocí *neuronové sítě*. *Neuronová síť* vykazovala vynikající výsledky, avšak s jednou nevýhodou. Příznakový vektor je tvořen tak, že ignoruje vnitřní částí objektů a předměty jsou rozpoznávány jen podle své vnější hrany.

Back - Propagation									
		parametry sítě				parametry Back-Propagation			
		skryté vrstvy				učení			
		počet neuronů v 1		počet neuronů v 2		počet kroků		chyba SSE	
		70		35		5000		0,000409	
						Doba učení		11,146s	
klasifikace									
scéna	objekt	1	2	3	4	5	6	7	8
7	rozpoz.	100%	100%	97%	100%	100%			
	skut.	✓✓✓	✓✓✓	✓✓✓	✓✓✓	✓✓✓			
	čas	0,07s							
	soubor	ScénaBP 7a.bmp							
8	rozpoz.	100%	100%	100%	97%	100%			
	skut.	✓✓✓	✓✓✓	✓✓✓	✓✓✓	✓✓✓			
	čas	0,1s							
	soubor	ScénaBP 8a.bmp							
9	rozpoz.	100%	100%	100%					
	skut.	✓✓✓	✓✓✓	✓✓✓					
	čas	0,08s							
	soubor	ScénaBP 9a.bmp							
10	rozpoz.	84%	74%	97%	100%	100%	86%	100%	100%
	skut.	✗	✓✓	✓✓✓	✓✓✓	✓✓✓	✓✓✓	✓✓✓	✗✗✗
	čas	0,141s							
	soubor	ScénaBP 10a.bmp							
11	rozpoz.	84%	74%	86%	100%				
	skut.	✗	✓✓	✓✓✓	✓✓✓				
	čas	0,091s							
	soubor	ScénaBP 11a.bmp							

Tab. 3 . Rozpoznávání neuronovou sítí

6.3 SYNTAKTICKÝ POPIS OBJEKTU

Ukázka syntaktického popisu objektu *Držák S* (obr. 6.5a) a navržená primitiva (obr. 6.5c).



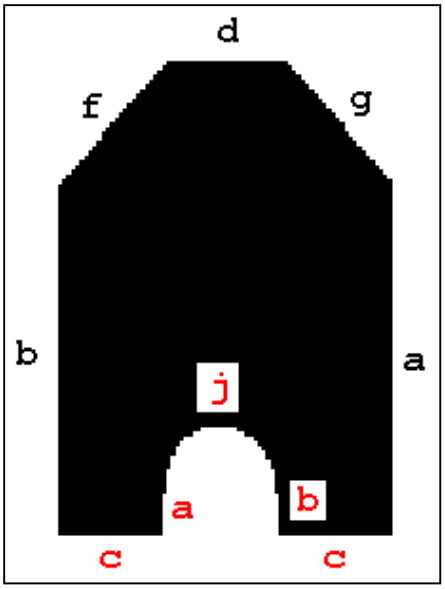
Obr. 6.5 Držák s detailem a navržená primitiva

V případě, že postupujeme od vyznačeného počátečního bodu (obr. 6.5a) proti směru chodu hodinových ručiček [38], pak dostaneme řetězec:

d f b c a j b c a g

Vlivem špatné kvality vstupního obrazu se může stát, že se nebudou detekovat krátké rovné úseky (obr. 6.5b) a tím se změní i řetězec objektu (obr. 6.6).

- d f b c a j b c a g
- d f b c j b c a g
- d f b c a j c a g
- d f b c j c a g



Obr. 6.6 Objekt s popisem alternativními řetězci

Pro tento případ je vhodné upravit gramatiku tak, aby generovala i alternativní řetězce (obr. 6.7).

Gramatika generující řetězec d f b c a j b c a g	Gramatika generující řetězce d f b c j b c a g d f b c a j c a g d f b c j c a g
S → dA A → fB B → bC C → cD D → aE E → jB E → g	S → dA A → fB B → bC C → cD C → cE D → aE E → jB E → jC E → g

Obr. 6.7 Úprava gramatiky pro alternativní řetězce

Tato forma přístupu je vhodná pro předpokládané chyby (deformace) při identifikaci daného objektu. V případě náhodných deformací je vhodné použít klasifikaci pomocí algoritmů pro zjištění vzdálenosti mezi atributovými popisy obrazů (kap. 5.3.1).

6.4 UKÁZKA VÝSLEDKŮ TESTOVÁNÍ SYNTAKTICKY POPSANÝCH OBJEKTŮ

Obr. 6.8 představuje ukázku rozpoznání testované scény metodou *syntaktické analýzy* a metodou *Levenshtein* z vytvořeného simulačního prostředí.

Obr. 6.8 Rozpoznání scény *syntaktickou analýzou* a metodou *Levenshtein*

V tab. 4 jsou výsledky klasifikace pomocí *syntaktické analýzy*. V případě použití primitiv označujících jednotlivé úseky hranic je *gramatika* citlivá na drobné chyby v detekci hran. *Gramatiku* je nutno sestavit „na míru“ pro určitý typ detektoru hran.

Syntaktická analýza					
Scéna	Čas [s]	Klasifikace			Výsledný soubor
		Správně	Chybně	Neident.	
1	0,26	2	0	3	Scene_01_Gram.bmp
2	0,23	0	0	5	Scene_02_Gram.bmp
3	0,27	2	0	3	Scene_03_Gram.bmp
4	0,14	1	0	4	Scene_04_Gram.bmp
5	0,13	0	0	3	Scene_05_Gram.bmp
6	0,09	1	0	2	Scene_06_Gram.bmp
7	0,121	0	0	3	Scene_07_Gram.bmp
8	0,08	2	0	1	Scene_08_Gram.bmp
9	0,31	1	0	4	Scene_09_Gram.bmp
10	0,16	1	0	4	Scene_10_Gram.bmp
11	0,291	1	0	4	Scene_11_Gram.bmp
12	0,17	2	0	3	Scene_12_Gram.bmp

Tab. 4 Výsledky testů pro *syntaktickou analýzu*

V tab. 5 jsou souhrnné výsledky pro metodu *Levenshtein.* Výsledné časy implementovaných algoritmů pro zjištění vzdálenosti mezi atributovými popisy řetězců získané z vytvořeného simulačního prostředí jsou pouze orientační – pro porovnání výsledků. Postupným výpočtem vzdáleností se totiž uchovává velké množství informací, které jsou potřebné pouze pro testování. Vynecháním testovacích informací při praktickém nasazení se čas pro výpočet algoritmů ještě sníží.

Levenshtein							
Scéna	Čas [s]	Klasifikace			Výsledný soubor	Hloubka	Percentil
		Správně	Chybně	Neident.			
1	0,13	3	1	1	Scene_01_Lev_0_70.bmp	0	70
2	0,16	3	1	1	Scene_02_Lev_0_70.bmp	0	70
3	0,12	3	1	1	Scene_03_Lev_0_70.bmp	0	70
4	0,15	3	1	1	Scene_04_Lev_0_70.bmp	0	70
5	0,141	2	0	1	Scene_05_Lev_0_70.bmp	0	70
6	0,05	3	0	0	Scene_06_Lev_0_70.bmp	0	70
7	0,11	2	0	1	Scene_07_Lev_0_70.bmp	0	70
8	0,05	3	0	0	Scene_08_Lev_0_70.bmp	0	70
9	0,241	2	0	3	Scene_09_Lev_0_70.bmp	0	70
10	0,11	3	0	2	Scene_10_Lev_0_70.bmp	0	70
11	0,221	2	0	3	Scene_11_Lev_0_70.bmp	0	70
12	0,09	3	0	2	Scene_12_Lev_0_70.bmp	0	70

Tab. 5 Výsledky testů pro algoritmus Levenshtein

7 ZÁVĚR

Přínos práce spočívá ve využití výsledků při řešení konkrétních aplikací. Bylo navrženo a implementováno původní simulační vývojové prostředí (ukázky viz obr. 6.4 a obr. 6.8) pro testování původně vytvořených algoritmů zejména pro syntaktické metody (ukázka kap. 5.3, kap. 6.3) pro rozpoznávání objektů technologické scény. Na základě simulačních experimentů v implementovaném prostředí byla provedena analýza výsledků (kap. 6) s ohledem na uvažované aplikační nasazení u průmyslových robotů.

S každou operací provedenou na scéně se snižuje počet informací, které jsou ve scéně obsaženy. Proto je vhodné, pokud je to možné (záleží na kvalitě pořízené scény), metody předzpracování obrazu vynechat.

Rozpoznávání klasickou *momentovou metodou* lze doporučit pro aplikace, kde není tolik důležitý přesný průběh hrany, ale postačí pouze „hrubé“ rozdělení do jednotlivých tříd. Např. nezáleží na tom, zda je mezi dvěma hranami ostrý přechod, nebo krátký oblouk. Momentová metoda je invariantní vůči natočení, ovšem za cenu pomalejšího výpočtu.

Rozpoznávání *neuronovou sítí* je vhodné pro případy, kde vyžadujeme vysokou rychlost klasifikace s libovolným natočením scény a příp. i tam, kde potřebujeme tolerovat určité rozdíly mezi naučenými etalony a klasifikovanou scénou.

Rozpoznávání objektů pomocí *syntaktické analýzy* je vhodné pro aplikace, kde je potřeba detekovat velmi přesně jednotlivé úseky hran, bez možnosti vzniku významné chyby a tam, kde je vyžadována vysoká rychlost klasifikace. Za významnou chybu je považována chyba, kterou nelze zahrnout do pravidel gramatiky.

Z algoritmů pro zjištění vzdálenosti mezi atributovými popisy obrazů byly na základě provedené analýzy implementovány dva nejvhodnější - *Levenshtein* a *Needleman-Wunsch.* Obě tyto metody, podle uskutečněných testů, poskytují přibližně stejné výsledky, které jsou velmi nadějné zejména s ohledem na případy, kdy nelze použít metodu syntaktické analýzy.

8 LITERATURA

[1] Blaška, J.: Moderní metody analýzy signálu, [online]. 1998. Dostupné z: <www.sh.cvut.cz/~blaska/>.

[2] Boef, J., Belin, P.: Fourier Descriptors, [online]. 1999. Dostupné z: <www-sig.enst.fr/tsi/enseignement/ressources/mti/descript_fourier/>.

[3] Bogr, J.: Rozpoznávání obrazů metodou Fourierových deskriptorů. [Disertační práce], *FE VUT*, Brno, 1984.

[4] Bogr, J., Holčík, J., Kozumplík, J.: Teorie diagnostiky biosystémů. *SNTL*, Praha, 1982.

[5] Cohen, W., Ravikumar, P., Fienberg, S.: Second String. *Carnegie Mellon University,* [online]. Dostupné z: <http://secondstring.sourceforge.net/>.

[6] Demel, J.: Grafy a jejich aplikace. *Academia*, Praha, 2002.

[7] De Juan, Ch.: Contour Recognition Problem, [online]. 2001. Dostupné z: <www.cc.gatech.edu/classes/ay2000/cs7495_fall/participants/cnd/ps3/ps3.html>.

[8] Fisher, R.: World and Scene Representations, [online]. 2002. Dostupné z: <www.dai.ed.ac.uk/CVonline/repres.htm>.

[9] Glynn, E. F.: Computer lab, [online]. 2002. Dostupné z: <www.efg2.com/lab>.

[10] Gonzales, R. C., Woods, R. E.: Digital Image Processing. *Addison-Wesley Publishing Co.*, 1993.

[11] Health, M., Sarkar, S.: Edge detection comparison, [online]. 1996. Dostupné z: <marathon.csee.usf.edu/edge/edge_detection.html>.

[12] Hipr: Image processing learning resources, [online]. 2000. Dostupné z: <www.dai.ed.ac.uk/HIPR2>.

[13] Hlaváč, V., Šonka, M.: Počítačové vidění. *Grada*, Praha, 1993.

[14] Holub, J.: Simulation of NFA in Pattern Matching. *ČVUT*, Praha, [online]. Dostupné z: <http://cs.felk.cvut.cz/~holub/>.

[15] Hopcroft, J. E., Ullman, J. D.: Formálne jazyky a automaty. *Alfa,* Bratislava, 1978.

[16] Chapman, S.: String Metrics, University of Sheffield, [online]. 2005. Dostupné z: <http://www.dcs.shef.ac.uk/~sam/stringmetrics.html>.

[17] Jelínek, I., Sochor, J.: Algoritmy počítačové grafiky. *ČSVTS*, Praha, 1989.

[18] Kepka, J., Psutka, J.: Kombinace příznakových a strukturálních metod rozpoznávání. *ZČU*, Plzeň, 1994.

[19] Khorosware: Edge detection I, [online]. 2000. Dostupné z: <www.ee.bgu.ac.il/~greg/graphics/special.html>.

[20] Kotek, Z., Mařík, V. a kol.: Metody rozpoznávání a jejich aplikace. *Academia*, Praha, 1993.

[21] Nazarion, J.: Libdistance, [online]. 2004. Dostupné z: <http://www.monkey.org/~jose/software/libdistance/>.

[22] Nilsson, N. J.: Principles of Artificial Intelligence. *Springer Verlag*, Berlin, 1982.

[23] Novák, M. a kol.: Umělé neuronové sítě. Teorie a aplikace. *C. H. Beck*, Praha, 1998.

[24] Pelikán, J.: Vyplňování souvislé oblasti. *MFF UK*, Praha, [online]. 2000. Dostupné z: <http//sun3.ms.mff.cuni.cz/~pepca/>.

[25] Pop, M.: Sobelova detekce hran, [online]. 2000. Dostupné z: <http://radio.fel.cvut.cz/courses/>.

[26] Prata, S.: Mistrovství v C++. *Computer Press*, Praha, 2001.

[27] Shaw, C. A.: Picture Graphs, Grammars and Parsing. In: Frontiers of Pattern Recognition. *Academic Press,* New York, 1972.

[28] Skála, V.: Algoritmy počítačové grafiky I–III. *ZČU*, Plzeň, 1992.

[29] Smith, M.W., Davis, W.A.: A new Algorithm for Edge Detection. 1974.

[30] Sochor, J.: Počítačová grafika. Brno, *MUNI FI*, [online]. 2002. Dostupné z: <http://www.fi.muni.cz/usr/sochor/>.

[31] Sochor, J., Žára, J., Beneš, B.: Algoritmy počítačové grafiky. *ČVUT*, Praha, 1998.

[32] Sviták, R.: Detekce hran v obraze. *ZU FAV*, Plzeň, [online]. 2001. Dostupné z: <http://herakles.zcu.cz/~rsvitak/school/zvi_rsvitak.pdf>.

[33] Šnorek, M., Jiřina, M.: Neuronové sítě a neuropočítače. *ČVUT*, Praha, 1998.

[34] Šonka, M.: Course Digital Image Processing, [online]. 2000. Dostupné z: <css.engineering.uiowa.edu/~dip>.

[35] Šonka, M., Hlaváč, V., Boyle, R.: Image Processing, Analysis and Machine Vision. *PWS*, Boston, 1998.

[36] Šťastný, J., Škorpil, V.: Image Processing by Using Neural Network. In: *International Scientific Conference of FME*, VŠB - TU Ostrava, 2000.

[37] Šťastný, J., Škorpil, V.: Analysis of Methods for Edge Detection. *International journal Communications 2003*, ISSN 0018-2028, 2003, 19 pp.

[38] Šťastný, J., Škorpil, V.: Comparison Methods for Pattern Recognition. *International Journal WSEAS Transactions on Circuits and Systems*. Issue 9, Volume 3, November 2004, ISSN 1109-2734, 6 pp.

[39] Šťastný, J., Škorpil, V.: Neural Network Learning Methods for Image Processing Applications. *International Journal WSEAS* , 2005, 6 pp.

[40] Uytterhoeven, G., Roos, D., Bultheel, A.: Wavelet Transforms Using the Lifting Scheme. *University Leuven,* Belgium, 1997.

[41] Uytterhoeven, G., Wulpen, F., Jansen, M., Roos, D., Bultheel, A.: Wavelets with Integer Lifting. *University Leuven*, Belgium, 1997.

[42] Žára, J., Beneš, B., Felkel, P.: Moderní počítačová grafika. *Computer Press*, Praha, 1998.

ABSTRACT

The habilitation thesis *Nontraditional Methods and Algorithms for Object Recognition of Technological Scene* deals with problems of computer image processing with a view to applications in the area of industrial robotics. The processing and recognition of the digitized image are solved in several following steps: image pre-processing, image segmentation, object description and pattern recognition.

Besides classical methods for object recognition especially nontraditional methods and algorithms (the Multi Layer Perceptron neural network with the Back-Propagation algorithm and the Levenshtein algorithm) were tested in simulation environment. Implemented algorithms were tested on the simulated objects of technological scene given.

The contribution of the work is the proposal and implementation of original simulation environment for testing objects of technological scenes, the implementation of algorithms for nontraditional methods for object recognition of technological scene and the analysis of simulation experiments with regard to on-coming applications at industrial robots control.

The results of the simulation experiments show that the Multi Layer Perceptron neural network with the Back-Propagation algorithm and the Levenshtein algorithm are very promising for object recognition of technological scenes for the use of industrial robots control.