Mendelova univerzita v Brně
Provozně ekonomická fakulta

Návrh systému pro analýzu 3D objektů pomocí deformačních gramatik

Diplomová práce

Vedoucí práce:
Prof. RNDr. Jiří Šťastný, CSc.

Bc. Lukáš Junek

Brno 2016

Rád bych poděkoval panu Prof. RNDr. Ing. J. Šťastnému CSc. za odborné vedení, za pomoc a rady při vypracovávání této práce.

Čestné prohlášení

Prohlašuji, že jsem práci: **Návrh systému pro analýzu 3D objektů pomocí deformačních gramatik**
vypracoval/a samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 23. května 2016 _____

Abstract

Junek, L. The system for the analysis of 3D objects through deformation grammars. Thesis. Brno: Mendel University in Brno, 2016.

This thesis deals with the design and implementation of systems for recognition of three-dimensional objects which are described by a two-dimensional array of distances.

First there is a description of the method of converting the input array of values to the terminal symbols. The thesis subsequently deals with some various grammars describing the object with its production rules. The aim is to find the simplest grammar as possible that does not include a large number of ambiguities. A parser processing of this grammar will work straightforward and faster.

Finally the parser working with a grammar describing the objects in the various descriptive levels is implemented. In conclusion the entire system is tested from the viewpoints of the accuracy of detection and the time that is required for analysis.

Keywords

Thesis, Structural methods, Object recognition, Matrix grammar, Deformation grammar, LR table, Tomita parser.

Abstrakt

Junek, L. Návrh systému pro analýzu 3D objektů pomocí deformačních gramatik. Diplomová práce. Brno: Mendelova univerzita v Brně, 2016.

Diplomová práce se zabývá návrhem a implementací systému pro rozpoznávání trojrozměrných objektů, které jsou popsány pomocí dvojrozměrného pole hodnot vzdáleností od středu snímání.

Nejprve je popsán způsob převodu vstupního pole hodnot na terminální symboly. Následně jsou představeny různé gramatiky popisující objekt pomocí produkčních pravidel. Snahou je nalézt co možná nejjednodušší gramatiku, která neobsahuje velký počet nejednoznačností. Díky tomu bude práce syntaktického analyzátoru přímočařejší a rychlejší.

Nakonec je implementován parser pracující s gramatikou popisující objekty v jednotlivých popisných rovinách. V závěru práce je celý systém otestován z hlediska správnosti detekce a času potřebného k analýze.

Klíčová slova

Diplomová práce, strukturální metody, rozpoznávání objektů, maticové gramatiky, deformační gramatiky, LR tabulka, Tomita parser.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod a cíl práce

Moje diplomová práce je zaměřena na rozpoznávání deformovaných 3D objektů pomocí deformační gramatiky. Jednotlivá tělesa budou popsána pomocí pole hodnot vzdáleností od středu snímání a mým úkolem bude navrhnout systém, který dokáže správně jednotlivé objekty rozpoznat. Klíčovým faktorem bude, kromě přesnosti, rychlost rozpoznávání, kdy převážně průmyslové provozy často vyžadují rozhodnutí o podobě objektu v reálném čase.

1.1 Úvod

S neustálým růstem výkonu výpočetní techniky se zároveň zvyšují nároky, jež jsou kladeny na nově vznikající systémy. Ne jinak je tomu v oblasti rozpoznávání objektů. V dnešní době, kdy dochází k masovému využití multimédií, se jednotlivé principy soustředí především na detekci objektů uvnitř obrazu.

Postupem času vznikají nejrůznější techniky pracující s širokou škálou parametrů jednotlivých objektů. Všechny se víceméně snaží nalézt takovou klíčovou vlastnost, která je unikátní pro jednotlivé posuzované prvky. Postup pří hledání odlišností a jejich následné zpracování se však často diametrálně liší.

Mezi jedny z nejužívanějších technik patří detekce na základě specifického tvaru hran objektů. Dalšími možnými metodami jsou detekce specifického tvaru klíčových částí objektu (popřípadě tvaru pomyslného obrazce vzniklého spojením těchto klíčových částí), postupné skládání výsledného objektu pomocí rozpoznávání dílčích částí objektu, analýza všech bodů nacházejících se na obraze a další.

Méně časté jsou již detekce objektů na základě jejich tvaru získaného odečtením klíčových bodů na povrchu objektu. Pro rozpoznání lze užít analýzy vzájemné polohy jednotlivých bodů, popřípadě posouzení velikosti dílčích úhlů mezi pomyslnými plochami.

V dnešní době se při detekci objektů stále více dostává do středu pozornosti čas potřebný pro úspěšné rozpoznání. Rostoucí počet systémů je nasazován do běžného prostředí a je od nich vyžadováno, aby své funkce plnily v reálném čase.

Příkladem může být požadavek identifikace jednotlivých osob zachycených pomocí kamerového systému, rozpoznání konkrétních poznávacích značek projíždějících automobilů, či úspěšné rozdělení komponent na výrobní lince a jejich přiřazení správnému zařízení.

V neposlední řadě je důležité si uvědomit, že většina zmíněných systémů postupem času nachází své uplatnění i v domácnostech a různých nástrojích sloužících obyčejným lidem v zaměstnání.

Charakteristickou vlastností těchto systémů je nutnost pracovat stejně dobře a spolehlivě i na zařízeních, které neslouží primárně pro rozpoznávání a tudíž neoplývají takovým výpočetním výkonem.

1.2 Cíl práce

Cílem mé diplomové práce je navrhnout a implementovat pomocí vhodného programovacího jazyka systém, který bude rozpoznávat trojrozměrné objekty pomocí strukturálních (syntaktických) metod. Zároveň je třeba zanalyzovat současné postupy sloužící k rozpoznávání objektů využívající tyto metody, jelikož z nich budu při svém návrhu vycházet.

Hlavním požadavkem na vytvářený systém je rychlost rozpoznávání objektů. V automatizovaných provozech je často klíčová schopnost jednotlivých zařízení rozhodovat v reálném čase (často do jedné vteřiny). Proto i mnou navržená aplikace by měla být schopná během tohoto času objekt správně zanalyzovat a rozpoznat.

Největší problém při rozpoznávání objektů často představuje nepřesné či deformované zaznamenání vstupního objektu. Dále tyto objekty mohou být různě natočené, případně mohou být i rozdílně velké. Je třeba vymyslet mechanismus, který si dokáže s těmito deformacemi poradit a objekt správně zařadit.

Vzhledem k tomu, že systém bude pracovat s trojrozměrnými objekty, vyvstává zde ještě jedna obtíž. Tyto objekty už nelze jednoduše popsat pomocí jednoduchých řetězců znaků (symbolů). Dvourozměrné objekty mohly být analyzovány pomocí znaků, kdy každý znak udává například směr, kterým hrana objektu vede. Takto definované objety mohly být popsány pomocí bezkontextových gramatik a pro jejich rozpoznávání lze využít metody pracující s těmito gramatikami.

V mém případě je nutné rozšířit popis objektů o třetí rozměr, díky čemuž nelze využít klasické bezkontextové automaty, ale je třeba použít složitější systémy. Další možností je určitým způsobem transformovat trojrozměrný popis tak, aby bezkontextové gramatiky byly použitelné.

2 Jednotlivé systémy pro rozpoznávání objektů

V následující kapitole se pokusím shrnout jednotlivé přístupy, které slouží k rozpoznávání objektů. Popíši, jakým způsobem fungují a na základě čeho se snaží dosáhnout úspěšné detekce.

2.1 Strukturální metody pro detekci objektů

Pro detekci objektu je možné užít například syntaktický přístup (1). V takovém případě jsou jednotlivé objekty popsány pomocí řetězce základních primitiv. Mezi základní primitiva je možné počítat například tvar hran objektu, klíčové úhly, které svírají jednotlivé části objektu, dále například samostatné dílčí části obrazu, které nejsou dále popsány (konkrétní tvar očí s konkrétní barvou).

Takovým popisem obrazu získáme slovo (řetězec terminálních symbolů). Pomocí derivačních pravidel se syntaktický analyzátor snaží sestrojit nad objektem konkrétní derivační strom. Předpokladem je, že každému objektu náleží právě jeden unikátní strom.

2.1.1 Typy gramatik

Množina slov popisující obrazy tvoří specifický jazyk. Jazyky lze obecně definovat pomocí výčtu všech jeho slov, nebo pomocí formálních gramatik (2). V případě výčtu lze však narazit na možnost, že jazyk je nekonečný. Z tohoto důvodu se příliš neužívá. Naproti tomu formální gramatiky generují slova pomocí množiny produkčních pravidel. Na základě tvaru pravidel rozlišujeme čtyři základní typy gramatiky (takzvaná Chomského hierarchie gramatik (3)).

- **Typ 0:** Neomezenou gramatiku Chomský definuje jako uspořádanou čtveřici $G = (N, T, P, S)$, kde: N je množina nonterminálů, T je množina terminálů, kdy platí $N \cap T = \emptyset$, P je konečná množina přepisovacích pravidel, které definujeme: $P \subseteq (N \cup T)^* N (N \cup T) \times (N \cup T)^*$ Nakonec S je počáteční symbol náležející do množiny nonterminálů (3).
- **Typ 1 (kontextová gramatika):** Kontextová gramatika zavádí určitá omezení na levé straně pravidla (2). Pokud $v \to w \in P$, pak platí, že $|v| \leq |w|$. Pravidla mají obvykle tvar $xAy \to xzy$, kde $A \in N$ a $x, z, y \in (\mathrm{N} \cup T)^*$. Symboly x, y udávají kontext, z tohoto důvodu nazýváme gramatiku kontextovou.
- **Typ 2 (bezkontextová gramatika):** Jak již název napovídá, bezkontextovou gramatiku lze vytvořit odstraněním kontextu (3). Vždy platí, že: $v \to w \in P$, kde $v \in N$ a $w \in (N \cup T)^*$ Bezkontextové gramatiky jsou v současnosti nejpoužívanější pro svoji popisnou sílu a jednoduchost zpracování pomocí zásobníkových automatů (4).

- **Typ 3 (regulární gramatiky):** Regulární gramatika zavádí další omezení pro možné tvary pravidel. Ta mohou mít tvar $A \rightarrow xB$ nebo $B \rightarrow x$, kde $A, B \in N$ a $x \in T^*$ (2). Regulární gramatiky se využívají především při konstrukci konečných automatů.

2.1.2 Rozšíření gramatiky o deformační pravidla

Při zpracování nedeformovaných objektů nelze očekávat velké problémy (5). Každý řetězec primitiv odpovídá právě jednomu objektu, který bude bez obtíží zpracován. Problém však mohou představovat (a obvykle představují) náhodné deformace těchto objektů. Zpracování deformovaných objektů bývá často obtížné, jelikož mohou vznikat náhodné řetězce, které nejsou popsány gramatickými pravidly.

Řešením může být rozšíření původní gramatiky o další pravidla popisující všechny možné chyby. Aby bylo možné odlišit nedeformované řetězce od ostatních, je třeba zvolit určitou metriku pro posuzování deformací, například lze použít měření vzdáleností deformovaných řetězců od svého vzoru (Hammingova vzdálenost (6), Euklidova vzdálenost (7) apod.).

Vznikne deformační gramatika $G' = (N', T', P', S)$, kde $N \subseteq N'$, $T \subseteq T'$ a $P \subseteq P'$. Rozšířené množiny tak obsahují kromě původních prvků i symboly, primitiva a pravidla popisující tyto deformace. Deformační pravidla jsou někdy navíc opatřena váhou (8), které udává, jak moc se část řetězce generovaná tímto pravidlem liší od předpokládaného tvaru.

Velké množství pravidel popisující velice obdobné tvary vstupních řetězců však často vede k přílišné komplexnosti a nejednoznačnosti celé gramatiky (5). Je tedy třeba upravit syntaktický analyzátor, aby byl schopen nedeterminismy určitým způsobem řešit.

2.1.3 Základní přístupy zpracování gramatiky

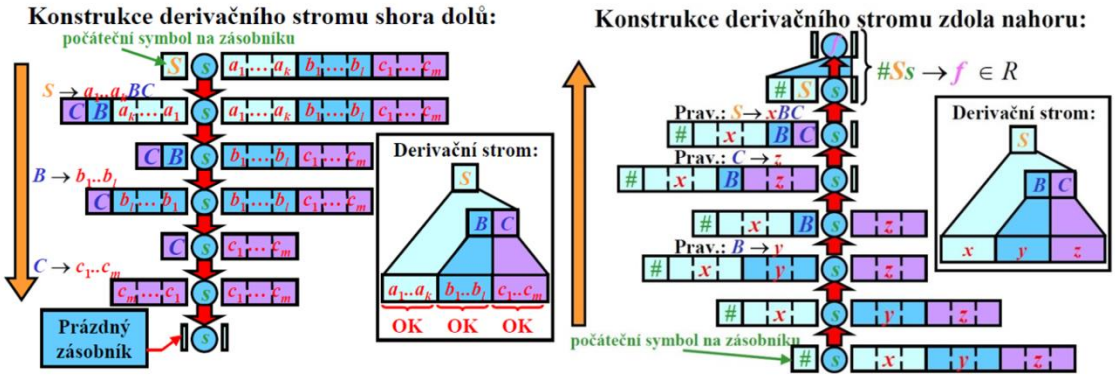
V dnešní době se hojně využívají dva základní přístupy pro konstrukci syntaktických analyzátorů zpracovávajících formální gramatiky (4) – analýza seshora dolů (top-down) nebo zespoda nahoru (bottom-up). Nejčastěji jsou zpracovávány bezkontextové gramatiky za použití jednoho zásobníku.

Obecné syntaktické analyzátory zpracovávající vstup **seshora dolů** aplikují na vstupní řetězec tzv. nejlevější rozpor. Je-li na vrcholu zásobníku terminální symbol, snaží se jej automat přiřadit aktuálnímu vstupnímu primitivu. Naopak je-li na vrcholu zásobníku nonterminál, je třeba jej derivovat pomocí určitého pravidla tak, aby pravá strana pravidla odpovídala vstupnímu řetězci (2).

Pomocí výše popsané metody se nejčastěji zpracovávají LL-gramatiky, které definují množiny FIRST, FOLLOW a EMPTY pomocí kterých je vytvořena rozhodovací LL tabulka. Díky tomu nemusí automat při určování derivačních pravidel prohledávat celý stavový prostor (4).

Top-down automat je poměrně jednoduchý na implementaci. Na druhou stranu je relativně slabý a jej nelze modifikovat pro zpracování deformačních nejednoznačných gramatik.

Syntaktické analyzátory pracující **zespod nahoru** pracují obráceně. Obdobně jako v předešlém případě dochází k rozdělení vstupu na nezpracované terminály a větnou formu obsahující jak terminály, tak nonterminály vzniklé redukčními pravidly. Automat provádí dvě základní akce SHIFT a REDUCE. Pomocí první operace nahrává terminály na zásobník, pomocí druhé pak na tyto terminály (a již redukované nonterminály) aplikuje derivační pravidla a redukuje je na nové nonterminály (2).



Obr. 1 Příklad principu fungování obou základních přístupů k syntaktické analýze
Zdroj: Přednášky do předmětu IFJ na FIT VUT Brno, 2009

Opět lze upravit gramatiku a vytvořit rozhodovací tabulku, která umožňuje automatu postupovat podle předem vytvořené posloupnosti stavů. Mezi nejznámější automaty patří Earlyho parser, Precedenční parser a LR parser. (4)

Bottom-up automaty jsou obecně silnější než top-down, navíc je možné upravit Earlyho parser nebo LR parser pro analýzu nejednoznačných deformačních gramatik.

2.1.4 Automaty zpracovávající deformační gramatiky

Nejednoznačná gramatika představuje pro většinu syntaktických analyzátorů problém. Nemají totiž implementovaný žádný mechanismus, který by si dokázal poradit v situacích, kdy není přechod ze stavu do stavu jednoznačně určen (5). Navíc automat musí akumulovat váhy deformačních pravidel, na základě kterých dojde k finálnímu verdiktu.

V dnešní době se při práci s nejednoznačnými gramatikami uplatňují modifikovaný Earlyho parser, Tomita parser (4) a LRE(k) parser (9) (hybrid mezi LR parserem a Earlyho automatem).

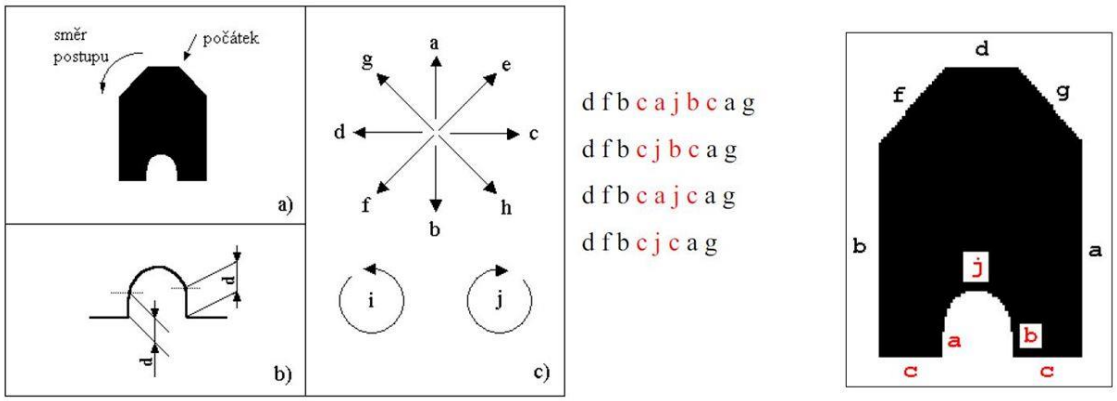
Earlyho parser v základu pracuje obdobně jako při zpracování jednoznačné gramatiky, liší se pouze způsob vytváření derivačního stromu, kterých může být více (5). Na Earlyho syntaktický analyzátor navázal LRE(k), který využívá pro svou činnost předpovědní LR tabulku, díky čemuž dokáže ovlivnit počet potenciálních stavů. Nakonec Tomita parser rovněž využívá ke své činnosti LR tabulku, nicméně prochází všechny možné stavy a nesnaží se z nich vybírat.

2.1.5 Rozpoznávání objektů na základě popisu směru jeho hran

K popisu jednotlivých objektů je možné využít Freemanovy řetězcové kódy (chain codes) (10). Tyto objekty jsou popsány znaky, z nichž každý reprezentuje určitý směr dílčí hrany objektu.

Existují dvě varianty řetězcových kódů – popisující hrany ve čtyřech směrech nebo ve směrech osmi (v tomto případě jsou řetězce symbolů kratší, neboť dokáží zaznamenat šikmé hrany jedním znakem) (1).

Pro zjištění rozdílů mezi jednotlivými deformovanými řetězci je možné využít metody meziřetězcových vzdáleností (Hammingova vzdálenost (6), Euklidova vzdálenost (7) apod.).



Obr. 2 Příklad popisu hran objektů pomocí řetězcových kódů i s příkladem znázornění možných deformací
Zdroj: Netradiční metody a algoritmy pro rozpoznávání objektů technologické scény, 2006 (1)

2.1.6 Detekce objektů na základě rozpoznání jeho obrazových částí

Na rozdíl od detekce objektu pomocí směru jeho dílčích hran, tato metoda provádí jeho rozpoznání na základě správného zjištění všech částí objektu. Předpokládá se, že všechny části daného předmětu jsou opět objekty, které se opět skládají z dalších předmětů (8). Například člověk se skládá z hlavy, trupu, končetin. Na hlavě lze dále rozpoznávat oči, ústa, nos apod.

Navíc jednotlivé díly mají své vlastní kategorie a podkategorie. Například podoba sedícího nebo stojícího člověka bude rozdílná, je možné dále upřesnit výraz v obličeji atd.

Kromě jednotlivých objektů popisuje gramatika také jejich pozici (protože například pozice ruky ovlivňuje sklon ramenou) a samotné umístění částí objektu uvnitř snímku.

Jednotlivé objekty a všechny jeho kategorie jsou popsány pomocí nonterminálů. Části celku jsou pak generovány pomocí produkčních pravidel. Pro popis předmětu je využita bag grammar, která je podobná bezkontextové gramatice, ale na rozdíl od ní generuje multi-množiny terminálů, tzv. bags (ne jejich řetězce).

Generující pravidlo je pak ve tvaru:

$$X \xrightarrow{\alpha} \{Y_1, \dots, Y_n\} \in P,$$

kde $X \in N, Y_i \in (T \cup N), \alpha \in R$. Nonterminály obvykle reprezentují větší celky objektu, popřípadě i celý objekt. Bags na pravé straně je množina všech vygenerovaných dílčích částí, případně primitiv. Řecké písmeno alfa (tzv. skóre) reprezentuje odchylku od předpokládané skutečnosti. Cílem je dosáhnout co možná nejvyšší skóre během celé syntaktické analýzy (8).

Množina Ω představuje všechny myslitelné pozice terminálů v obrazu. Umístění jednotlivých primitiv na obraze je možné zapsat jako Y(ω), kde $Y \in N \cup T, \omega \in \Omega$. Pravidla pak mají tvar:

$$X(\omega_0) \xrightarrow{\alpha} \{Y_1(\omega_1), \dots, Y_n(\omega_n)\} \in P.$$

Schéma umístěných redukčních pravidel má následující formu:

$$\forall z \in D: X(\omega_0(z)) \xrightarrow{a(z)} \{Y_1(\omega_1(z)), \dots, Y_n(\omega_n(z))\}.$$

D je množina parametrických redukčních pravidel definovaných schématem, $z \in D$ definuje umístěná redukční pravidla a $\omega_i(z)$ jsou funkce udávající pozice v Ω. Řecké písmeno alfa opět představuje skóre (8).

Například redukční schéma pro obličej je:

$$\forall(\omega_0, \omega_1, \omega_2, \omega_3) \in \Omega^4: FACE(\omega_0) \xrightarrow{\alpha(\omega_0,\omega_1,\omega_2,\omega_3)} \{LEFT.EYE(\omega_1), RIGHT.EYE(\omega_2), MOUTH(\omega_3)\}$$

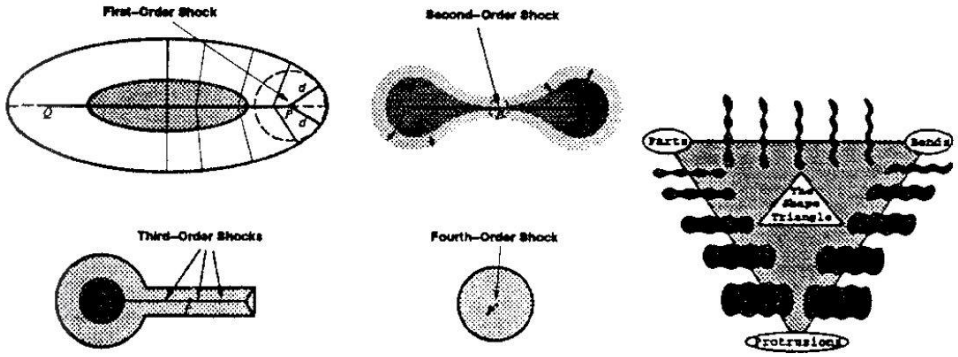
2.1.7 Šoková gramatika pro popis objektů

Šoková gramatika si klade za cíl rozpoznat jednotlivé objekty za pomocí tzv. šoků neboli zvláštností, které vznikají při deformaci hranic předmětu (11). Šoky představují určité vzorce, které nevnikají náhodně, ale dodržují gramatická pravidla a navíc mají specifické topologické a geometrické vzorce. Hypotéza šoků vychází z toho, že porušení zákonitostí gramatiky jsou geometricky nebo topologicky nepřípustná a vedou k porušení globální rovnováhy.

Jednotlivá pozorování jsou organizována do hierarchických grafů skupin šoků počítaných v reakčně rozptýleném prostoru, kde rozptýlení hraje roli v určení příslušnosti k jednotlivé skupině šoků. Tyto skupiny pak funkčně náleží jednotlivým částem objektů, jeho výčnělkům nebo ohybům.

Při výstavbě operátorů pro detekci šoků, je třeba se vyrovnat se dvěma výzvami – klasifikací šoků a jejich detekcí (11). Intuitivně mohou být šoky popsány vlastnostmi hraničních bodů, nicméně toto přímé přiřazení je náročné na imple-

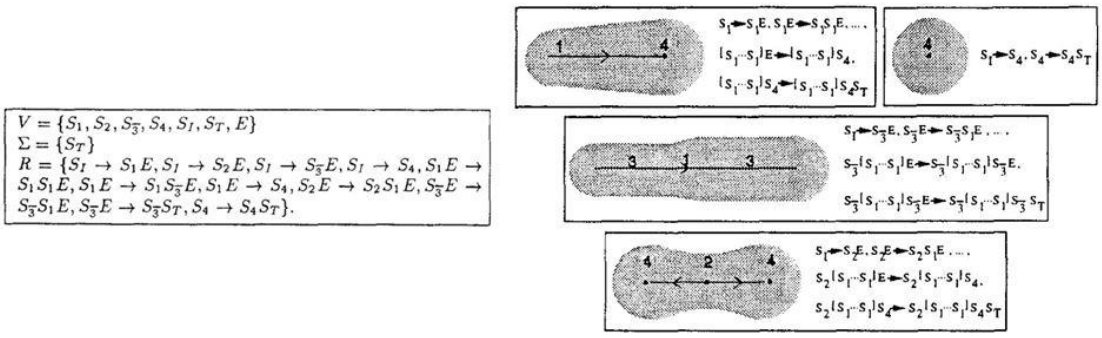
mentaci. Proto je pro tento účel využito vlastností vložených povrchů. Prvotní tok křivek je vložen do úrovně vývoje hodnoceného povrchu, přičemž nultá úroveň vždy reprezentuje odvozený povrch.



Obr. 3 Ukázka jednotlivých úrovní šoků
Zdroj: A Shock Grammar For Recognition, 1996 (11)

Pro účely detekce šoků byla vytvořena subpixelová implantace získávající přesný geometrický propočet v sousedství nespojitostí (11). Základní ideou je vybrat ze dvou spojitých množin bodů pro interpolaci jeden bod, který má nejnižší variaci. Náhradou polynomů geometrickými primitivy (jako přímky, oběžníky) můžeme postup adaptovat na problém nalezení úrovně křivek na vloženém povrchu.

V šokové gramatice pak nonterminály představují první, druhou, čtvrtou úroveň narušení a konec růstu šoku (zbavuje šoky kontextu). Terminálem je pak šok třetí úrovně. Produkční pravidla popisují šíření šoku na hranicích objektu.



Obr. 4 V levém rámečku je ukázka prvků šokové gramatiky a v pravém pak ukázka postupu aplikace derivačních pravidel
Zdroj: A Shock Grammar For Recognition, 1996 (11)

2.1.8 Stochastická obrazová gramatika

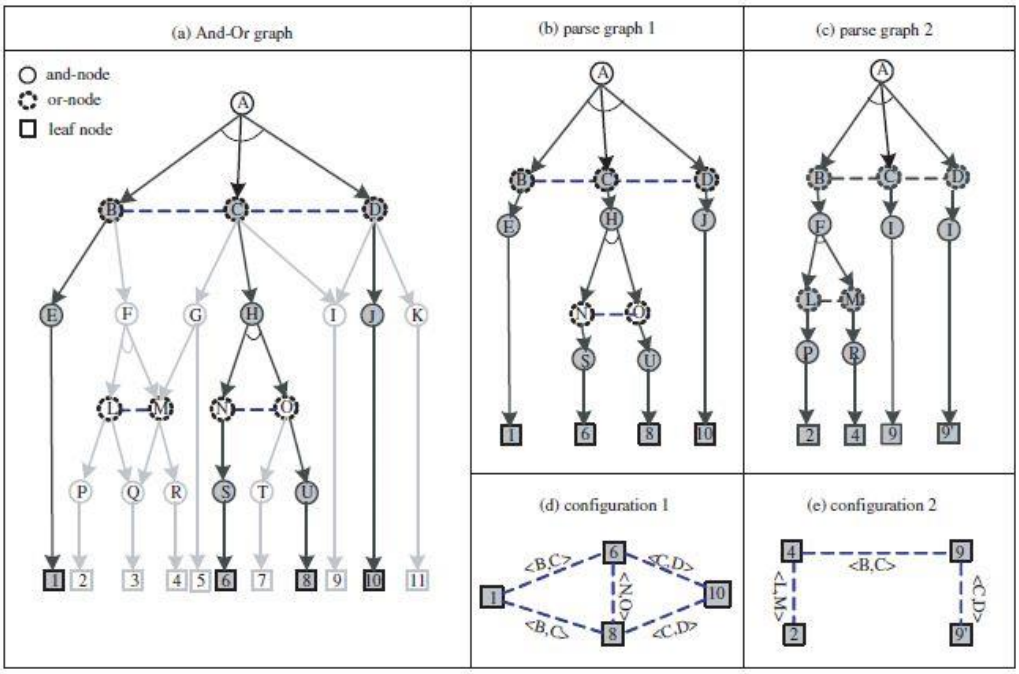
Stochastická obrazová gramatika představuje hierarchickou dekompozici scény na objekty (12), části objektů, primitiva a pixely (terminály a nonterminály), zároveň udržuje funkční relace mezi uzly pomocí horizontálních propojení. Každý objekt je definován jako množina všech možných konfigurací, které mohou být generovány gramatikou.

Gramatika může být zobrazena jako jednoduchý AND/OR graf, kde OR uzly představují alternativní pod-konfigurace a AND uzly jsou následně dekomponovány do určitého počtu komponent. Tato procedura pak podporuje rekurzivní top-down/bottom-up procedury pro zpracování obrazu.

Vzhledem ke vstupnímu obrazu, metoda vytvoří nejpravděpodobnější předběžný graf (12). Tato výstupní reprezentace následně představuje podgraf AND/OR grafu potom, co jsou učiněny všechny výběry v OR uzlu.

Pravděpodobnostní model je definován jako počet výskytů objektu a jeho částí, stejně tak jejich relací. To umožňuje provést učení pouze z relativně malé učící množiny pro jednotlivé kategorie. Díky zobecnění je pak možné pokrýt velké množství různých objektů.

Nakonec je v gramatice obsažen vizuální slovník, který pokrývá přechod mezi samotnými signály a gramatickými primitivy. Slovník je možné si představit jako množinu grafických primitiv, které obsahují sérii kotev, kterými se mapují na analyzovaný obraz.



Obr. 5 Ilustrace AND/OR grafu a jeho vtělení do gramatiky a jejích produkčních pravidel
Zdroj: A Stochastic Grammar Of Images, 2006 (12)

V grafové reprezentaci čtverce představují terminály, kulaté uzly pak neterminály. Každý AND uzel představuje dekompozici objektu na jeho části. To můžeme popsat pomocí gramatických pravidel $A \rightarrow BCD$ nebo $H \rightarrow NO$ (12).

Horizontální propojení mezi dětmi AND uzlu představují relace a omezení. OR uzel funguje vlastně jako přepínač pro alternativní podstruktury. Současně se může vyskytovat na více úrovních (kategorie scény, třída objektů a částí…). Pravidly je možné OR uzel zapsat jako $B \rightarrow E/F$ nebo $C \rightarrow G/H/I$.

Díky rekurzivní definici je pak možné jednotlivé části obrazu popsat samostatným grafem. Jednotlivé podgrafy nakonec vytvoří výsledný rozsáhlý AND/OR graf.

2.1.9 Maticová gramatika generující obrazce

Pro posouzení obrazů a jejich zpracování je možné využít i generativní modely pro zpracování digitálních obrazových polí ve dvoudimenzionálním prostoru (13). Tato dvojrozměrná pole lze posuzovat například pomocí maticových gramatik generujících dvojdimenzionální jazyky, jejichž základními primitivy jsou obrazová pole. Ke zpracování obrazu jsou často využívané hexagonální pole a hexagonální vzorce.

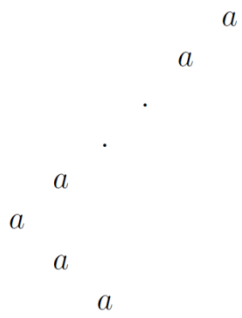
Takto konstruované gramatiky pak vycházejí z klasických gramatik a přejímají jejich základní techniky pro zvýšení generativní kapacity produkčních pravidel. Jednou z hlavních úprav maticových gramatik byla jejich přeměna na gramatiky s náhodným kontextem. Takto definované gramatiky se staly prototypem pro další editace a nakonec z nich vzešly bezkontextové gramatiky.

Dále je možné k pravidlům připojit omezující symboly, které představují jistý kontrolní mechanismus a zároveň dovolují širší zkoumání generativní síly gramatiky.

Maticovou gramatiku lze definovat jako pětici $G = (N, T, S, P, \#)$. N představuje neterminály, T terminály, $S \in N$ reprezentuje počáteční symbol, P je množina derivačních pravidel a nakonec $\#$ je prázdný symbol (13). Pravidla mají tvar $\alpha \rightarrow \beta$, kde α a $\beta \in N \cup T \cup \{\#\}$ představují v případě hexagonální gramatiky hexagonální podvzorce. Alfa a beta musí splňovat následující podmínky:

- Tvar, jejž reprezentují oba symboly, musí být totožný.
- α obsahuje alespoň jeden neterminál (aby byla gramatika bezkontextová, musí se na levé straně pravidla nacházet pouze jeden neterminál a nic jiného).
- Terminály a neterminály v α nejsou nahrazeny prázdnými symboly #.
- Aplikace produkčního pravidla udržuje konektivitu pole.

Například uvažujme gramatiku G = ({S,A,B},{a},P,S,#), která má produkční pravidla $P = \left\{ S \begin{matrix} \# \\ \# \end{matrix} \rightarrow a \begin{matrix} A \\ B' \end{matrix};\ A \begin{matrix} \# \\ \end{matrix} \rightarrow a \begin{matrix} A' \\ \end{matrix};\ B \begin{matrix} \\ \# \end{matrix} \rightarrow a \begin{matrix} \\ B' \end{matrix};\ A' \rightarrow A; B' \rightarrow B; A \rightarrow a\ ; B \rightarrow b \right\}$, vygeneruje vzorec na Obr. 6.



Obr. 6 Tvar vygenerovaný pomocí maticové gramatiky
Zdroj: Cooperating distributed context-free hexagonal array grammar systems with permitting context, 2014 (13)

2.2 Další metody pro detekci objektů

V následném textu se zaměřím na některé další možné metody vhodné pro popis, detekci a rozpoznání objektů, které však již nevyužívají strukturální principy.

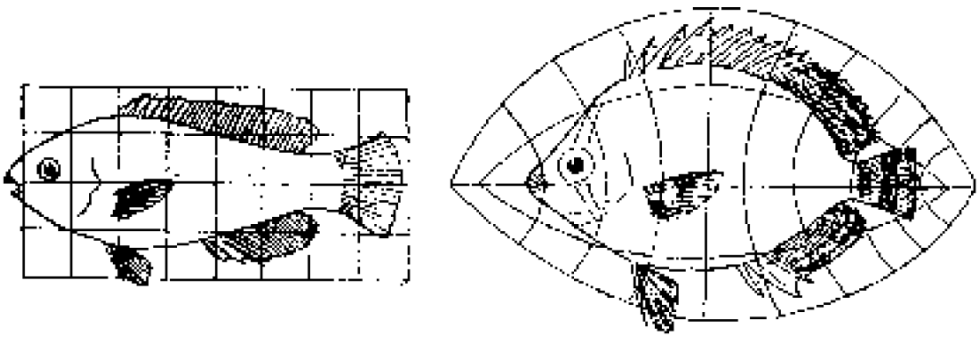
2.2.1 Rozpoznání obrazu na základě jeho tvaru

Metoda se zakládá na měření podobnosti mezi dvěma tvary (14). Podobnost je posuzována na základě shody mezi dvěma body různých tvarů. Body jsou získávány z kontur obrazu (většinou se jedná o množinu pixelů určujících hranu objektu). Posouzením shody bodů je následně možné odhadnout transformaci obou tvarů.

Pro určení podobnosti byla vytvořena míra kontext tvaru, která se nalézá v každém bodě. Kontext tvaru v klíčových bodech zachycuje distribuci ostatních bodů, které jsou vzhledem ke klíčovým bodům v relativní pozici.

Shodné body na dvou rozdílných tvarech mají obdobný tvarový kontext, což umožňuje převést úlohu na problém optimálního rozprostření. Nalezení shody lze považovat za ekvivalentní úlohu k nalezení dalšího bodu co nejvíce podobného bodu původnímu. Maximalizace podoby a vnucení unikátnosti vede k obdobnosti s úlohou hledání shody dvou biparitních grafů.

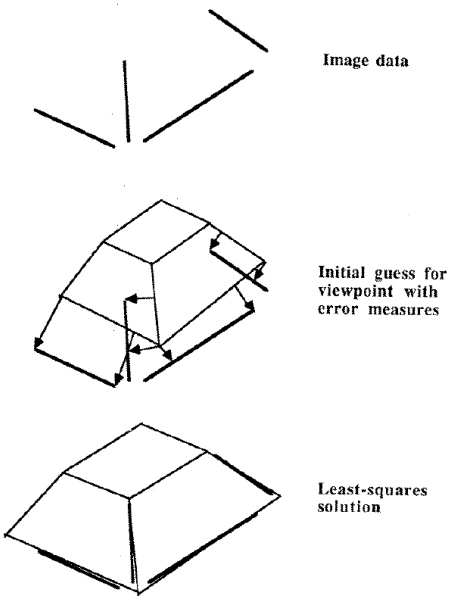
Míra shody je pak vypočítána jako suma chyb přiřazení mezi oběma obdobnými body. Tato chyba potom určuje rozsah transformace obou tvarů.



Obr. 7 Příklad obdobnosti tvarů při souřadnicové transformaci
Zdroj: Shape Matching and Object Recognition Using Shape Context, 2002 (14)

2.2.2 Odvození trojdimenzionálního objektu z obrazové informace

Cílem této techniky je rozpoznání trojdimenzionálních objektů v černobílém obraze (15). Principem této metody není snaha o rekonstrukci hloubky informace pomocí botom-up přístupu, naopak využívá dalších mechanismů, které dokáží zjistit informace o 3D objektech z dvojrozměrného obrazu.



Obr. 8 Princip projekce 3D tělesa k obrazovým datům
Zdroj: Three-dimensional object recognition from single two-dimensional images, 1987 (15)

Nejprve je proveden proces vizuální organizace, který vede k vytvoření shluků a struktur uvnitř obrazu (15). Ty jsou invariantní vůči změně úhlu pohledu a měřítka. Pravděpodobnostními metodami následně dojde k omezení počtu objektů,

mezi nimiž dojde k finálnímu přiřazování. Nakonec prostorovým přiřazením dojde k projekci jednotlivých trojrozměrných těles na příslušné místo v obraze.

Klíčovou částí celého procesu je najít úvodní propojení dvourozměrného obrazu a 3D těles. Tato propojení je pak možné vnímat jako určitá omezení, která musí všechny potenciální shody splňovat. Na základě těchto úvodních omezení je pak možné definovat další omezující podmínky a zjednodušit tak budoucí predikci.

2.2.3 Neuronové sítě

Primární výhodou užití neuronových sítí je (1), že není třeba znát postup řešení problému. U případu řešených touto metodou se nejčastěji vychází z trénovací množiny dat, kdy známe přesné výsledky určitých případů (avšak může dojít i k případům učení bez učitele).

Neuronové sítě se snaží napodobit fungování neuronů v mozku živočichů, kdy jsou tyto buňky provázány do vzájemných sítí a tyto sítě buněk při řešení problému vzájemně spolupracují. V IT neurony představují funkční prvek, který přijímá určitý počet signálů v podobě číselných hodnot, které transformuje na výstupní hodnoty.

Sítě pracují ve dvou režimech – v učícím a aktivním (1). Během učící fáze jsou stanoveny výchozí hodnoty parametrů, následně je prováděna samotná predikce.

Pro potřeby učení se často používá trénovací množina (tzv. učení s učitelem), kdy je dopředu znám výsledek predikce. Cílem této fáze je minimalizovat jednotlivé parametry organizované v maticích tak, aby nákladová funkce byla co nejnižší (tedy aby odchylky v předpovědích u učících dat byly co možná nejmenší).

Pro řešení složitějších problémů se užívá vícevrstevných neuronových sítí, které obsahují jednu nebo více skrytých vrstev. S počtem vrstev pak roste přesnost celé sítě, na druhou stranu může dojít až k přeučení, kdy se nepodaří vystihnout správný trend.

Neuronové sítě se často využívají ve spojení s jinými metodami, kdy pomáhají řešit klasifikační problémy.

3 Návrh systému pro rozpoznávání trojrozměrných objektů

V této kapitole se budu zabývat samotným návrhem systému pro rozpoznávání trojrozměrných objektů. Popíši jednotlivé možné přístupy, jak daný problém řešit, a také problémy, které s sebou tyto přístupy nesou. Nakonec rozpracuji nejvhodnější řešení a na jeho základě implementuji samotný systém.

3.1 Návrh objektů a systém jejich popisu

Vlastnosti mnou navrženého systému budu testovat na předem zvolené množině testovacích trojrozměrných objektů. Vybral jsem taková tělesa, pomocí kterých budu moci ověřit funkčnost systému a otestovat jeho vlastnosti.

Objekty by měly být co nejvíce různorodé. Pokud by měly stále stejné vlastnosti, existuje nebezpečí, že bych některé klíčové vlastnosti budoucího systému dostatečně nerozpracoval.

Dále je třeba určit způsob popisu těchto 3D primitiv. Textové soubory popisující jednotlivé objekty budou představovat samotný vstup pro rozpoznávání. Vzhledem k třetímu rozměru nelze tělesa popsat jedním řetězcem, neboť ten nedokáže reflektovat v dostatečné míře třetí rozměr objektu.

3.1.1 Navržené objekty

Jako vstupní objekty jsem v převážné většině zvolil základní trojrozměrná primitiva, jako je krychle/kvádr, koule, osmistěn (dvojitý jehlan), obrácený dvojitý jehlan apod. Tato tělesa představují do určité míry jisté zjednodušení oproti skutečnosti. Především jsou často relativně symetrické a jejich tvar je daleko jednodušší oproti objektům reálného světa.

Přesto stále mají potřebné vlastnosti společné všem objektům. Jejich povrch se skládá z jednotlivých ploch a hran, které tyto plochy svírající určitý úhle oddělují. Při zpracování těles například v počítačové grafice se při vnější reprezentaci objektů rovněž pracuje s tělesy v podobě polygonů, tedy množiny rovných ploch oddělených hranami.

Základní geometrická tělesa obsahují takových ploch méně. Způsob zpracování ale zůstane stejný. Jediným rozdílem je, že nebudu muset použít větší počet popisných bodů a vzniknuvší výsledná gramatika tak bude do určité míry jednodušší. Přesto bude potřeba pro její zpracování pokročilých syntaktických analyzátorů.

Výchozí 3D primitiva jsem doplnil o dva složitější tvary v podobě váz, na kterých hodlám demonstrovat schopnost systému rozpoznat i složitější objety.

3.1.2 Princip popisu objektů

Systém bude na svém vstupu předpokládat dvojrozměrné pole hodnot, kde každá hodnota představuje vzdálenost povrchu tělesa od osy snímání, která prochází svisle středem objektu.

Předpokladem je, že každé těleso je snímáno v několika vodorovných rovinách aspoň ze čtyř úhlů, které rovnoměrně popisují daný objekt. Jednotlivé řádky představují vzdálenosti povrchu tělesa pro všechny snímané úhly v jedné rovině, sloupce pak vzdálenosti pro všechny roviny ve snímaném úhlu.

Těleso může být obecně snímáno z volitelného počtu úhlů v libovolném počtu rovin. Aby však byl objekt korektně popsán, jsou vždy vyžadovány aspoň tři vodorovné roviny a čtyři popisné úhly, které dohromady obsáhnou všech 360°.

V mé práci budu pracovat se snímáním z osmi úhlů, kdy každý úhel je násobkem 45 stupňů. Počet rovin pak může být proměnlivý, vždy jich však musí být lichý počet. Prostřední rovina totiž představuje střed tělesa, kde střed této roviny je zároveň v počátku souřadného systému.

Tab. 1 Příklad vstupního pole pro dvojitý jehlan

0°/360°	45°	90°	135°	180°	225°	270°	315°
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
1,414	1,000	1,414	1,000	1,414	1,000	1,414	1,000
2,828	2,000	2,828	2,000	2,828	2,000	2,828	2,000
4,242	3,000	4,242	3,000	4,242	3,000	4,242	3,000
2,828	2,000	2,828	2,000	2,828	2,000	2,828	2,000
1,414	1,000	1,414	1,000	1,414	1,000	1,414	1,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000

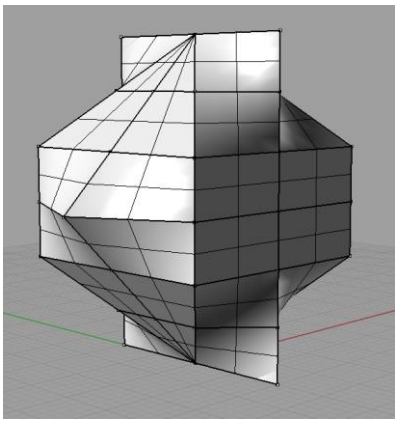
Z výše uvedených skutečností vyplývá, že objekty jsou v podstatě popsány jednotlivými body vždy v určitých úhlech a určitých rovinách. To však vede k určité deformaci celého 3D tělesa. To je totiž snímáno vždy v pevných místech, které se nepřizpůsobují tvaru objektu. Téměř nikdy se proto nepodaří rozmístit body ideálně na kraje hlavních ploch (tedy přímo na hranu). Navíc se změnou velikosti objektu se úhly mezi jednotlivými plochami mění, protože postupně přibývají jednotlivé meziplochy.

Pro nepootočená základní 3D tělesa nepředstavuje tato deformace problém, protože mají hrany umístěny ve stejných místech, jako jsou rozmístěny snímající body. Odchylka proto je takřka neznatelná. Deformace se však zásadně projevuje při otočení těles. Ani 3D primitiva pak už nemusí mít (a často skutečně nemají) hrany umístěny v klíčových a mnou snímaných úhlech. Objekty jsou proto popsány značně nepřesně, neboť systém zaznamená pro různé úhly natočení pokaždé jinak rozmístěné body. Na Obr. 9 je vidět deformace pro otočený kvádr.

Celá situace se drobně liší od obdobných strukturálních problémů. Tyto systémy rozpoznávají nedeformované objekty, jež jsou však vlivem nepřesného práce

snímací kamery chybně popsány. V mém případě jsou jednotlivé údaje správně zaznamenané, deformace vznikají díky nedostatečnému počtu zpracovávaných bodů na povrchu tělesa, kdy dochází ke ztrátě informace.

Pokud bychom přidali další body, je pravděpodobné, že by došlo alespoň k částečné eliminaci většiny deformací. Úloha by se však mohla stát příliš rozsáhlou. Zároveň by klesla rychlost rozpoznávání, jež je pro systémy rozpoznávání často velice klíčová.



Obr. 9 Příklad deformace kvádru otočeného o 45° kolem osy *z*

3.1.3 Omezení vstupních těles

Výše popsaný způsob popisu vstupních těles má však některá omezení. Předně je nutné, aby povrch tělesa v každém úhlu pohledu neprotínal osu snímání tělesa, jež leží ve středu samotného objektu. Případně, aby s ní neměl společné body (existuje výjimka, pokud se jedná jen o jeden bod – např. koule má v horní a spodní rovině vždy jeden bod právě na ose tělesa). Dále je nutné, aby těleso obepínalo osu po celé výšce. Z důvodu, aby se nevyskytovaly záporné hodnoty vzdáleností.

Posledním omezením, které využiji ve své práci, je úhel otočení objektů. Jednotlivé objekty budou moci být otočeny vždy pouze o 45 stupňů v libovolné ose, přičemž otočené objekty musí splňovat výše definované podmínky o bodech mimo osu snímání.

3.2 Popsání tělesa pomocí maticové gramatiky

Pokud by systém zpracovával dvojrozměrná primitiva popsaná jednoduchými znaky odpovídajícími hranám, ve vstupním souboru by se nacházel pouze samotný řetězec znaků. Ten by nebylo třeba dále upravovat. Stačí, když je jednotlivým znakům přiřazen příslušný terminál. Ty pak syntaktický analyzátor dále zpracovává a utváří z nich derivační strom.

V mém případě je však situace o poznání komplikovanější. Není totiž jasné, které terminály odpovídají kterým hodnotám vzdáleností. Pro 2x zvětšený objekt budou vzdálenosti rovněž větší, přičemž by jim měl odpovídat stejný terminál. Ne-

dá se tedy bezpečně určit, jaký terminál přísluší dané hodnotě. Celé pole hodnot tedy budu muset napřed převést do vhodnějšího systému, jenž bude umožňovat jednoznačné přiřazení terminálů ke svým prvkům.

3.2.1 Návrh maticové gramatiky

Těleso je popsáno pomocí bodů na jeho povrchu. Pokud dojde ke spojení sousedních bodů pomyslným úsečkami, je možné tímto způsobem získat trojúhelníkové (platí pro situace, kdy vzdálenosti v jedné rovině popisují právě jeden bod) nebo čtyřúhelníkové plochy definované těmito body. Jednotlivé úsečky pak tvoří hrany mezi těmito plochami. Získáme pomyslnou polygonovou síť, která se běžně užívá při zpracovávání objektů v počítačové grafice.

Těmto základním primitivům (plochám a hranám) lze již jednoznačně přiřadit terminály a vytvořit tak maticovou gramatiku. Pomocí derivačních pravidel postupně bude možné vygenerovat z počátečního symbolu jednotlivé objekty složené z těchto ploch a hran.

V případě 2D obrazce skládajícího se z ploch a hran by byl celý postup dostačující, protože se těleso rozkládá pouze v jedné rovině. Pro trojrozměrná tělesa však obvykle platí, že dvě plochy mezi sebou svírají i určitý úhel. Aby bylo možné tento úhel získat, je nutné nejprve převést hodnoty vzdáleností na hodnoty souřadného systému ve 3D prostoru.

3.2.2 Převedení vstupního pole na množinu bodů v prostoru

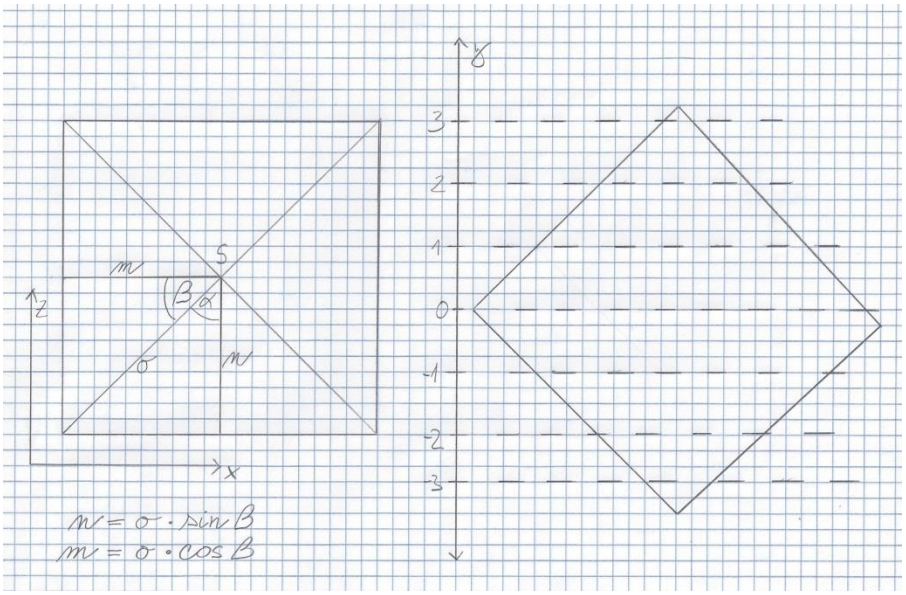
Změnou hodnot vzdáleností na hodnoty souřadného systému získáme obdobný vnější popis objektu pomocí krajních bodů. Podstatným rozdílem však bude, že v tomto případě již lze využít analytické geometrie pro výpočet úhlů mezi jednotlivými plochami. Pro převod bodů určených jednotlivými hodnotami vzdáleností na body souřadného systému je možné použít standardní goniometrické funkce.

Nejjednodušší je výpočet hodnoty na ose *y*. Těleso je vždy popsáno lichým počtem řádků představující jednotlivé vodorovné roviny. Prostřední řádek pak představuje rovinu ve středu tělesa. Pokud tuto rovinu posuneme do středu souřadného systému, kdy osa tělesa bude procházet bodem (0, 0, 0), můžeme na základě této skutečnosti konstatovat, že všechny body ve středové rovině mají hodnotu na ose *y* rovnu nule. Hodnota *y* bodu v dalších řádcích (rovinách) se pak jednoduše dopočítají podle pořadí daného řádku oproti řádku střednímu.

Šlo by zvolit ještě jednodušší způsob výpočtu hodnoty *y*, kdy první řádek by odpovídal hodnotě rovné nule a všechny další řádky by pak měly hodnotu o jedna menší (nebo větší) oproti řádku předchozímu. Nicméně užitím výše zmíněného způsobu se vyhnu některým problémům při dalším zpracovávání (viz kapitola 3.2.3).

Hodnoty na osách *x* a *z* lze dopočítat goniometrickými funkcemi *sinus* a *cosinus*. Jako úhel pro výpočet použijeme vždy hodnotu úhlu, z něhož byl snímán objekt v daném sloupci. Abych dostal očekávanou hodnotu, je nutné k tomuto úhlu

připočíst dalších 45 stupňů (první sloupec je snímán z úhlu 45°). Určíme-li vzdálenost bodu od osy snímání jako velikost přepony, pak hodnotu na ose *x* můžeme určit jako velikost přilehlé odvěsny a hodnotu na ose *z* jako velikost protilehlé odvěsny.



Obr. 10 Způsob výpočtu bodů v souřadném systému z hodnot vzdáleností

Je-li proveden tento výpočet pro všechny hodnoty vzdáleností, dostanu na konci výpočtu pole bodů v trojrozměrném prostoru. V tomto okamžiku je již možné určit směrové a normálové vektory všech ploch utvořených spojením sousedních bodů.

3.2.3 Výpočet úhlů mezi plochami

Pro výpočet jednotlivých úhlů mezi všemi plochami lze využít vzorců z analytické geometrie, konkrétně vlastnosti normálových vektorů ploch. Ty jsou vždy kolmé vzhledem k rovině plochy. Úhel mezi normálovými vektory dvou ploch tak odpovídá úhlu, který dvě roviny mezi sebou svírají.

Pro každou plochu vyberu tři body, ze kterých utvořím směrové vektory. Ty definují danou plochu. Pomocí těchto směrových vektorů lze vypočíst vektorovým součinem odpovídající normálový vektor plochy. Rovněž není nic složitého vypočítat úhel mezi dvěma normálovými vektory ploch pomocí skalárního součinu.

Pro celý proces je klíčové, aby výběr bodů pro výpočet směrových vektorů byl pokaždé stejný. V opačném případě se může stát, že vypočtený normálový vektor bude o 180 stupňů otočený. Došlo by tak k určitému zkreslení vypočteného úhlu mezi normálovými vektory.

Abych dostal pro horní a spodní polovinu stejný počet ploch, je vhodné postupovat s výběrem směrových vektorů pro každou polovinu tělesa zrcadlově obráceně. Jinak dojde k vypuštění jednoho řádku ploch těsně pod vertikálním středem tělesa. Zároveň díky tomu získám stejné velikosti úhlů pro symetrická tělesa

v obou polovinách, čímž opět mírně zjednoduším gramatiku popisující tělesa. Tato skutečnost mě ostatně vedla k výběru složitější varianty při výpočtu hodnoty souřadnice *y*.

V některých případech je přeci jen nutné pozměnit způsob výpočtu směrových vektorů, respektive upravit způsob výběru bodů pro výpočet. Pokud je ve vstupním poli vzdáleností v každém sloupci dané roviny nulová hodnota, pak vlastně v každém sloupci popisuje tato hodnota jeden stejný bod objektu.

V okamžiku, kdy by při výběru bodů pro výpočet směrového vektoru byly zvoleny dva body z tohoto řádku, vznikly by dva stejné směrové vektory. Normálový vektor pak bude nulový a celý výpočet úhlů mezi plochami chybný. V takové situaci je vhodné způsob výpočtu směrových vektorů upravit. Je však nutné zkontrolovat výsledný směr normálového vektoru a případně jej přepočítat do požadované pozice.

3.2.4 Konkávní tvar objektů

Samotný výpočet úhlu mezi normálovými vektory plochy však není dostačující. Jedna plocha se k druhé může přiklánět nebo se naopak od ní odvracet. Skalární součin sice poskytuje informace o úhlu mezi dvěma vektory, avšak o jejich vzájemné poloze nic neříká.

Pro určení vzájemné polohy vektorů ve sloupci je možné využít pomocného vektoru *v*, který je rovnoběžný s osou snímání. Nezbytnou podmínku pro určení správného náklonu plochy představuje nutnost umístit oba normálové vektory a pomocný vektor do stejné roviny. Této pozice vstupních normálových vektorů lze docílit pomocí 3D transformace, kdy otočíme oba vektory kolem osy *y* o příslušný úhel. Pro určení tohoto úhlu můžeme využít další pomocný vektor *u*, tentokráte vedoucí kolmo na osu snímání.

Nyní lze bez problému vypočítat vzájemné úhly mezi vektorem *v* a oběma normálovými vektory. Pokud je úhel mezi prvním normálovým vektorem a pomocným vektorem větší než v případě druhé dvojice, pak se druhá plocha k první přiklání, v opačném případě, kdy je úhel menší, se naopak plochy od sebe odklání.

Pro určení vzájemné polohy obou ploch v řádku však podobnou techniku, jaká byla zmíněna výše, využít nemohu, neboť se směry normálových vektorů mění v závislosti na kvadrantu, tudíž pomocný vektor mířící jen jedním směrem není řešením. Naopak pro otočení vektorů do jedné roviny není třeba využívat transformace, stačí pouze přiřadit za hodnotu *y* obou vektorů nulu (dojde tak k zobrazení obou vektorů ve 2D prostoru definovaném osami *x* a *z*).

Vzájemnou pozici dvou sousedních ploch je možné stanovit na základě posouzení velikostí hodnot *x*, respektive *z* obou normálových vektorů ploch v závislosti na kvadrantu. Pokud je hodnota *x* (respektive *z*) druhého normálového vektoru vyšší, plochy se od sebe odklání, v druhém případě se k sobě přiklání.

3.2.5 Doplnění horních a spodních ploch

U některých objektů, například neotočeného kvádru nebo konkávního válce, jsou z důvodu formátu vstupního pole hodnot vzdáleností „skryty“ horní a spodní plochy (podstavy), které jsou kolmé na osu snímání.

V těchto případech jsem přidával pro každý sloupec dva nové řádky, které obsahují trojúhelníkové plochy a hranu o úhlu 90°. Nepřidání těchto ploch a hran by nepředstavovalo velký problém, rozdíly by se daly popsat pomocí gramatiky. Přidání zbývajících ploch má tak spíše kosmetický efekt a učinil jsem tak pro úplnost.

Daleko významnější efekt bude mít doplnění těchto hran v gramatice popisující tvar těles, viz kapitola 3.5.5.

3.2.6 Vstupní pole ploch pro maticovou gramatiku

Výsledkem operací popsaných v kapitolách 3.2.3 a 3.2.6 je pole jednotlivých ploch a hran mezi těmito plochami. Každá hrana navíc obsahuje informaci, jaký úhel svírají dvě plochy, mezi kterými leží. Nyní je tak již možné sestavit maticovou gramatiku, pomocí které budou jednotlivá tělesa popsána.

Příklady vstupních polí (již převedených na terminály) je možné spatřit v Tab. 3 a v Tab. 4.

3.2.7 Problém rotačních ploch

Posledním problémem před definováním maticové gramatiky, jenž je třeba vyřešit, je existence rotačních ploch. Vstupní objekt může být díky malému počtu popisných bodů značně deformovaný. Pokud mají protilehlé hrany určité plochy odlišný sklon, potom mezi čtyřmi sousedními body vznikají rotační plochy. Příklad takové deformace je patrný na Obr. 9.

V takovém případě se vypočtený normálový vektor vždy odlišuje pro každou kombinaci vstupních směrových vektorů získaných z krajních bodů plochy. Úhly mezi plochami získané skalárním součinem těchto vektorů tak mohou být dosti náhodné.

Rotační plochy vznikají především v místech hran, které nejsou díky deformaci vstupního tělesa (tedy díky malému počtu popisných bodů) zaznamenané. V drtivé většině případů body vypočítané ze vstupních hodnot vzdáleností, které tvoří rotační plochu, leží na dvou různých plochách tělesa. Ty odděluje nezaznamenaná hrana, kdy jedna plocha je rovnoběžná s osou snímání (dva sousedící body v jednom sloupci mají stejnou vzdálenost od středu).

Řešení může představovat doplnění chybějící hrany a rozdělení této rotační plochy na dvě samostatné části. Vhodnou volbou směrových vektorů dostaneme dva normálové vektory, které korespondují se skutečným tvarem tělesa. Nevýhodu představuje skutečnost, že se může v každém řádku lišit počet vstupních ploch.

3.3 Definování prvků maticové gramatiky

Maticová gramatika G = (S, N, T, P, #), kde N je množina nonterminálů, T množina terminálů, S ∈ N je počáteční symbol, P je množina produkčních pravidel a # je speciální prázdný symbol.

Jednotlivým vstupním plochám a hranám nyní mohu přiřadit odpovídající terminální symboly z množiny T. Celý princip rozpoznávání trojrozměrných objektů bude probíhat stejně jako v případě jednoduchých 2D obrazců. Pro objekty na vstupu se budu snažit sestrojit příslušný derivační strom. Na základě sestrojení tohoto stromu pak rozhodnu, o jaký objekt se jedná.

3.3.1 Přiřazení vhodných terminálů plochám

Pro mnou navržený systém zpracovávající trojrozměrné objekty mohu počítat s převážně čtyřúhelníkovými plochami. Jen v některých případech vzniká plocha trojúhelníková (v případech, kdy jeden řádek má samé nulové vzdálenosti a popisuje tak jeden bod).

Budu tedy potřebovat dva terminální symboly odlišující tyto dvě plochy. V dalším textu budu používat symbol *x* pro čtvercovou plochu a symbol *y* pro plochu trojúhelníkovou.

U ploch není potřeba jakkoliv pracovat s jejich velikostí, protože rozpoznávání bude probíhat na základě samotného tvaru. Předpokladem je, že drtivá většina objektů má určitý specifický tvar. Rozdílná velikost stejných objektů má sice vliv na počet jednotlivých ploch a hran, celkový tvar zůstává ve většině případů přibližně stejný. Navíc pro rozpoznávání objektů není potřeba provádět sémantické operace, kde by rozměry plochy mohly být potřeba.

3.3.2 Přiřazení terminálů hranám

O něco složitější je přiřazení terminálů hranám. Plochy totiž mezi sebou mohou svírat obecně jakýkoli úhel. Přiřazení terminálního symbolu každému myslitelnému úhlu by bylo zbytečně složité. Gramatika popisující tyto terminály by s největší pravděpodobností neobsahovala velké množství nejednoznačností ztěžující rozpoznávání. Unikátní hrany budou derivovány pokaždé jinými nonterminály, takže výsledný syntaktický analyzátor bude mít velké množství neprovázaných stavů. Nicméně sestrojení systému zpracovávající tuto gramatiku by bylo značně komplikované a pracné.

Řešením může být přiřazení jednoho symbolu všem úhlům v určitém intervalu. Zde je klíčová volba velikosti intervalů, pokud by byly příliš malé, nedošlo by k zásadnímu zjednodušení gramatiky a přetrval by výše popsaný problém. Naopak při volbě příliš malého počtu intervalů se gramatika může stát velice nejednoznačnou.

V takovém případě bude činnost syntaktického analyzátoru prodlužovat nutnost procházet všechny možné stavy a zjišťovat, který je správný. Navíc v nejhorším případě může jeden derivační strom odpovídat více objektům a celý systém rozpoznávání tak bude nefunkční.

Při pohledu na deformace vstupních těles je možné zjistit, že vhodnou volbou intervalů je možné odstranit značné množství těchto deformací. Klíčové plochy trojrozměrných objektů mezi sebou svírají určitý úhel, který je logicky pro libovolně natočený objekt vždy stejný. Díky deformacím na vstupu se však může tento úhel pro zpracovávaný objekt určitým způsobem lišit. Nikdy však nejde o nikterak zásadní rozdíl.

Stanovením rozsahu intervalu pro všechny tyto možné velikosti deformovaných hran je tedy možné popsat tyto hrany stejným terminálem. Tím značně snížím nejednoznačnosti v gramatice, neboť bude silně zredukován počet derivačních pravidel nutných k popisu objektů a jejich hran.

Tab. 2 Volba terminálů vzhledem k úhlům mezi plochami vstupních objektů

terminál	interval
a	(5°; 40°>
b	(40°; 50°>
c	(51°; 85°>
d	(85°; 95°>
e	(95°; 130°>
f	(130°; 140°>
g	(140°; 180°)
h	(-5°; -40°>
i	(-40°; -50°>
j	(-50°; -85°>
k	(-85°; -95°>
l	(-95°; -180°)
m	<-5°; 5°>, 180°

Terminály *a-g* popisují uhly mezi plochami, kdy se obě tyto plochy od sebe odklání, naopak symboly *h-l* popisují úhel pro přiklánějící se plochy. Většina těles pak nemá příliš velké úhly mezi přiklánějícími se plochami, proto mohu použít pro úhly nad 90° už jen jediný terminál.

Při pohledu na Tab. 2 je vidět, že intervaly úhlů jsou uzpůsobeny vstupním objektům. Většina z nich jsou základní trojrozměrná tělesa, jejichž plochy svírají mezi sebou převážně pravé úhly nebo jejich polovinu.

Deformacemi na vstupu je často způsobeno, že dvě dílčí plochy tvořící jednu hlavní plochu mezi sebou svírají určitý malý úhel. Je proto vhodné terminálu *m* přiřadit rovněž určitý interval namísto nulové hodnoty.

Hodí se rovněž připomenout, že každá hrana popisuje úhel mezi normálovými vektory. Úhel, jejž svírají plochy uvnitř tělesa, se od vypočítaného úhlu liší. Lze ho však dopočítat jednoduchým přepočtem – vnitřní úhel = 180° - úhel mezi vektory.

Tab. 3 Příklad vstupního pole převedeného na terminály pro neotočený kvádr

y		y		y		y		y		y		y		y	
d		d		d		d		d		d		d		d	
x	m	x	d	x	m	x	d	x	m	x	d	x	m	x	d
m		m		m		m		m		m		m		m	
x	m	x	d	x	m	x	d	x	m	x	d	x	m	x	d
m		m		m		m		m		m		m		m	
x	m	x	d	x	m	x	d	x	m	x	d	x	m	x	d
d		d		d		d		d		d		d		d	
y		y		y		y		y		y		y		y	

3.3.3 Stanovení vhodných nonterminálů

Nonterminály by měly vhodně derivovat jednotlivé části objektů. První skupina nonterminálů bude derivovat samotné terminály, druhá pak jednotlivé části objektů.

Především určení druhé skupiny nonterminálů je klíčové. Do jednoho nonterminálu je třeba redukovat vždy klíčové části objektu. Tyto části se však díky deformacím na vstupu mohou určitým způsobem odlišovat. Pokud je však derivuje vždy jeden nonterminál, zjednoduší se tím popis objektu ve vyšších úrovních abstrakce. To pak povede ke snížení počtu pravidel.

Na druhou stanu je jasné, že redukce více možných verzí stejné části objektu do jednoho totožného nonterminálu povede s největší pravděpodobností k nejednoznačnostem, neboť vznikne velké množství obdobných pravidel. Velký počet stejných pravidel zvýší pravděpodobnost, že totožné pravé strany pravidel budou derivovány různými nonterminály. Vznikne tak nedeterministická gramatika, kterou nebude možné zpracovat jednoduchými syntaktickými analyzátory.

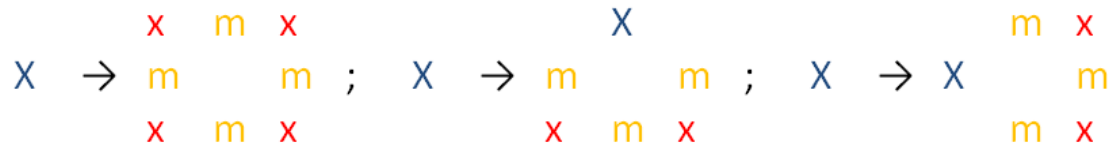
3.3.4 Určení derivačních pravidel

Při rozpoznávání deformovaných objektů je možné postupovat tak, že napřed sloučím všechny plochy ležící v jedné rovině do velké agregované plochy. Stejně je možné postupovat i pro hrany ležící mezi jednotlivými plochami. Pokud ty leží mezi plochami v jedné rovině, je možné je úplně vypustit, ve druhém případě je rovněž sloučím do jedné velké hrany.

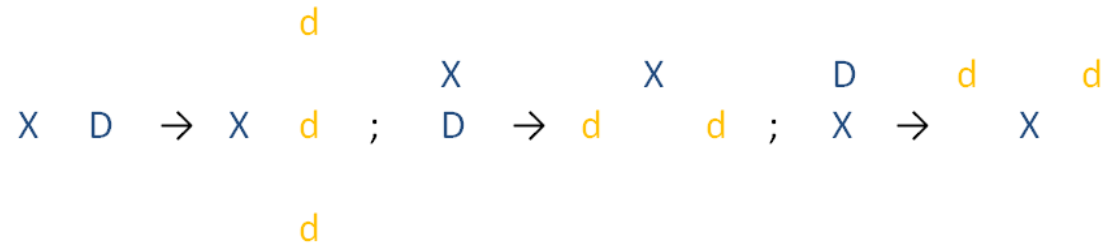
Jako příklad jsou uvedena derivační pravidla pro rozpoznání kvádru. Na Obr. 11 jsou pravidla pro sloučení menších čtvercových ploch do jedné a na Obr. 12 jsou příslušná pravidla pro sloučení hran. Nakonec připojíme vrchní a spodní trojúhelníkové plochy pomocí derivačních pravidel na Obr. 13.

V uvedených příkladech chybí na pravé straně pravidla prázdné symboly. Ty se vyskytují na místech, kam budou pomocí pravidla derivovány symboly nové. Vynechal jsem je z důvodu lepší čitelnosti pravidla, navíc pro nastínění způsobu popisu tělesa nejsou zas tak důležité. Díky tomu nejsou pravidla ani uvedena

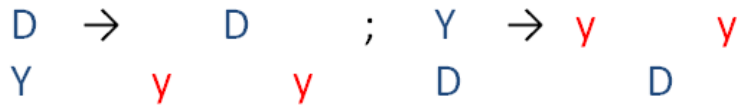
v bezkontextové formě, kdy z prázdných symbolů jsou napřed derivovány neterminály a až potom terminální symboly.



Obr. 11 Pravidla pro sloučení více ploch do jedné agregované plochy



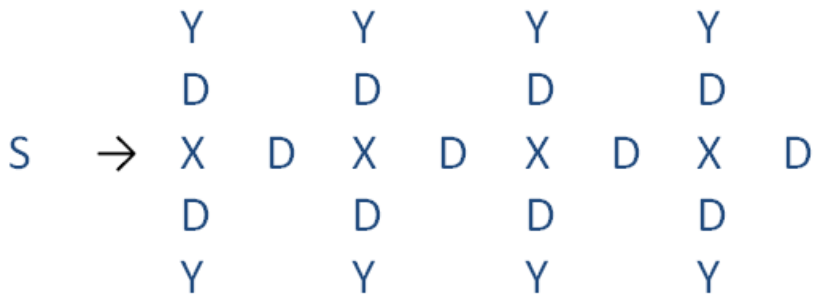
Obr. 12 Pravidla pro sloučení menších hran



Obr. 13 Pravidla pro derivování vrchních a spodních trojúhelníkových ploch

Postupně je možné agregovat jednotlivé prvky na hlavní plochy a hrany trojrozměrného objektu, které v podstatě odpovídají rozloženému plášti. Tyto vzniklé nonterminály je dále možné popsat pomocí obdobného principu. Na Obr. 14 je znázorněna varianta derivačního pravidla, kdy je celý objekt derivován z počátečního symbolu pomocí jednoho pravidla. Nic však nebrání použití i mezipravidel, která by derivovala napřed jen části objektu.

Rovněž zde nejsou pravidla zaznamenána přesně. Některá pravidla by měla derivovat kromě jiných prvků i prázdné symboly, jež potom budou nahrazeny dalšími objekty. Princip je však stále stejný.



Obr. 14 Pravidlo derivující agregované plochy z počátečního symbolu

3.3.5 Rozpoznávání otočených objektů

Situaci poněkud komplikuje možnost, že vstupní objekty budou při snímání různě pootočené. Vstupní pole terminálů se pak logicky bude výrazně lišit. Pro otáčení kolem osy *y* nepředstavuje tato skutečnost velký problém. Stačí postupně měnit pořadí sloupců, kdy postupně posouváme první sloupce na konec pole, dokud se nedostanu do výchozího stavu.

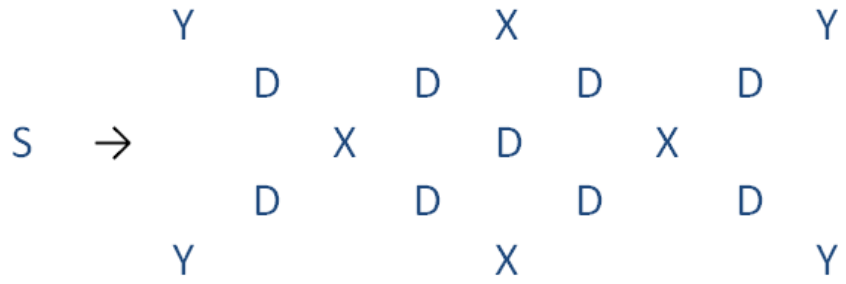
Komplikovanější je otočení kolem ostatních os. V takovém případě už prohazování pořadí sloupců nepomůže, neboť se klíčové hrany nacházejí v jiných místech. Tato změna nelze řešit jinak než přidáním dodatečných derivačních pravidel postihujících tyto proměny. Celá gramatika tak bude o něco komplikovanější.

Protože vstupní pole musí splňovat určité podmínky, nemusí gramatika popisovat všechny případy otočení, což na druhou stranu představuje určité ulehčení.

Tab. 4 Příklad vstupního pole terminálů pro kvádr otočený o 45° kolem osy z

x	m	y	d	y	m	x	m	x	m	x	m	y	d	y	m
m		d	y	d		m		m		m		d	y	d	
y	d		m		d	x	m	x	m	x	d		m		d
d	y	m	x	m	y	d		d		d	y	m	x	m	y
y	d		m		d	x	m	x	m	x	d		m		d
m		d	y	d		m		m		m		d	y	d	
x	m	y	d	y	m	x	m	x	m	x	m	y	d	y	m

Derivační pravidla pro popis otočených objektů budou mít stejný tvar jako základní pravidla popsaná v kapitole 3.3.4. Pravidlo derivující nonterminály z počátečního symbolu pro otočený objekt popsaný v Tab. 4 je znázorněno na Obr. 15.



Obr. 15 Pravidlo derivující nonterminály z počátečního symbolu pro kvádr otočený o 90°

3.4 Syntaktický analyzátor pro maticovou gramatiku

Nyní, když je maticová gramatika navržená, mohu sestrojit syntaktický analyzátor, který sestavením derivačního stromu rozpozná samotný objekt. Při pohledu na jednotlivá pravidla se ukazuje, že bude obtížné využít standardní syntaktické analyzátory pracující s přechodovou tabulkou.

Tyto analyzátory pracují s vždy s aktuálním znakem na vstupu a aktuálním prvkem na zásobníku. V tomto případě však není na první pohled zřejmé, které znaky na vstupu by se měly posuzovat první. Jednotlivá pravidla totiž popisují proměnlivé počty řádků a sloupců. Řešení může představovat užití obecných parserů nebo úprava stávajících systémů pro potřeby maticové gramatiky.

Obecné automaty pracují s prohledáváním vstupního prostoru, buď do šířky, nebo do hloubky. Snaží se vždy najít úsek vstupních dat, na který pasuje libovolné pravidlo. Potom metodou pokus-omyl na tyto úseky postupně aplikují derivační pravidla. V případě, že se dostanou do slepé uličky, vrací se o několik kroků zpět a zkouší další možnosti, dokud nakonec neuspějí.

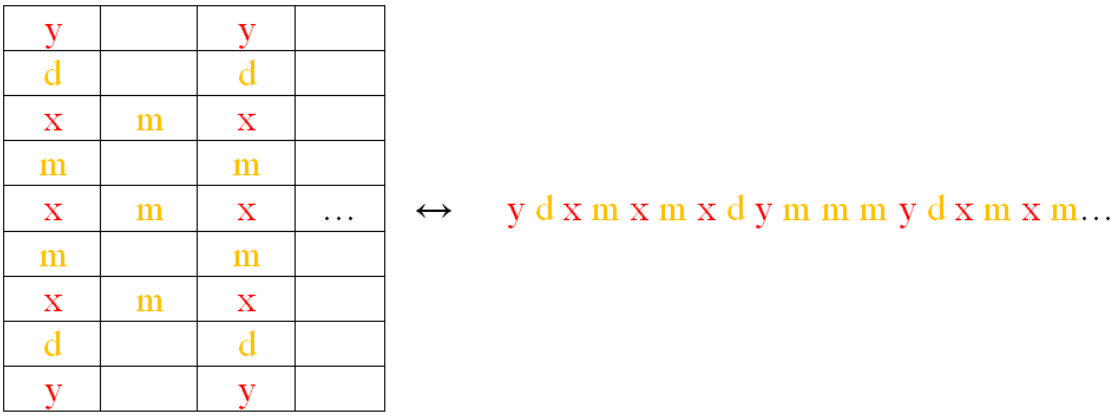
Je jasné, že prohledávání všech možností v daném prostoru není nejvhodnější a hlavně nejrychlejší. Vzhledem k tomu, že můj systém by měl umět rozpoznávat objekty v reálném čase, je třeba navrhnout lepší řešení.

Úprava stávajících systémů se zdá jako schůdnější. Dá se však očekávat velká pracnost tvorby tohoto automatu. Proto je vhodné zamyslet se nad možností úpravy samotné gramatiky. Pokud by bylo možné převést derivační pravidla do formy umožňující zpracování již existujícími syntaktickými analyzátory, představovalo by to značné zjednodušení problému.

3.4.1 Převod na bezkontextovou gramatiku

V této situaci je možné využít vlastností maticových gramatik. Ty se při dodržení určitých pravidel dají převést na standardní bezkontextové gramatiky. Přičemž mezi hlavní zásadu patří, že z neprázdného symbolu nesmí vzniknout symbol prázdný. Při pohledu na gramatiku definovanou v kapitole 3.3 je zřejmé, že se derivační pravidla spíše podobají kontextové gramatice než bezkontextové.

Pravidla je tedy nutné určitým způsobem upravit. Prozkoumáním vstupního dvojrozměrného pole ploch a hran je možné si uvědomit, že každé takové pole je určitým způsobem převeditelné na jednorozměrné, které je tvořeno sekvencí po sobě jdoucích řádků nebo sloupů. Podobným způsobem zpracovávají pole některé programovací jazyky, kdy sice k hodnotám v poli přistupujeme pomocí několika indexů, ale v paměti je pole uloženo jako posloupnost hodnot. Celý postup je znázorněn na Obr. 16.



Obr. 16 Způsob převodu dvojrozměrného vstupního pole na řetězec terminálů pro kvádr

Budeme-li číst vstupní pole po jednotlivých řádcích nebo sloupcích dostaneme na vstup klasickou posloupnost terminálních symbolů. Pro tuto posloupnost je pak jednoduché vytvořit standardní bezkontextovou gramatiku. Pro standardní bezkontextovou gramatiku je již možné navrhnout syntaktický analyzátor, který využívá pro svou práci rozhodovací tabulky.

Tvar jednotlivých pravidel je možné znázornit na příkladu neotočeného kvádru na Obr. 17. Automat postupně zpracovává jednotlivé sloupce a snaží se na vstupní symboly postupně aplikovat jednotlivá pravidla. Zpravované sloupce redukuje do větších celků, dokud nezpracuje celé pole.

Obr. 17 Příklad bezkontextové gramatiky pro neotočený kvádr

3.4.2 Nejednoznačná bezkontextová gramatika

Syntaktický analyzátor rozpoznávající deformované objekty na základě maticové gramatiky převedené na bezkontextovou gramatiku bude bezpochyby funkční.

Vzhledem k deformacím na vstupu, které je třeba popsat pomocí dalších derivačních pravidel, můžeme předpokládat, že tato gramatika bude pravděpodobně nejednoznačná. Navíc velké množství variant vstupních polí pro různě natočené objekty tuto skutečnost ještě více posílí.

Nedeterminismy prakticky vylučuje užití jednoduchých top-down nebo bottom-up parserů. Pro potřeby zpracování nejednoznačných gramatik je možné užít upravené LR syntaktické analyzátory (Tomita parser, Fast Early parser), které s nejednoznačnou gramatikou dokáží pracovat.

Tyto přístupy využívají ke své práci LR tabulku, na základě které rozhodují o následném stavu automatu, jenž bude následovat po stavu aktuálním. S tím však souvisí další obtíže.

Velké množství pravidel znamená, že sestrojit celou tabulku bude značně komplikované. Jde především o vysokou pracnost při její implementaci a hlavně obtížné odladění celého systému. Proto je nutné se zamyslet, zda by nešlo celý systém využívající upravenou maticovou gramatiku zredukovat a zjednodušit.

3.5 Gramatika popisující tvar objektu

V kapitole 3.2 jsem navrhl gramatiku popisující jednotlivé plochy a hrany, které jsem získal úpravou vstupního pole hodnot na pole bodů v souřadném systému. Následně jsem spojil sousední body a vytvořil pomyslnou polygonovou strukturu.

Celý způsob popisu těles se sice nakonec ukázal jako funkční, ale příliš komplikovaný. Předpokládaná gramatika by obsahovala velké množství nejednoznačností, jež komplikují sestavení příslušného syntaktického analyzátoru.

Cílem je tedy vymyšlení takové gramatiky, která dokáže spolehlivě popsat jednotlivá tělesa, zároveň bude jednodušší a navíc bude opět převeditelná na bezkontextovou gramatiku, díky čemuž budu moci využít standartních analyzátorů pracujících s rozhodovacími tabulkami.

3.5.1 Zjednodušení maticové gramatiky

Maticová gramatika popisovala všechny hrany a plochy. Při bližším pohledu je možné zjistit, že tvar jednotlivých těles je v převážné většině definován pouze samotnými hranami. Plochy představují „pouze“ rovná místa mezi těmito hranami. Navíc v drtivé většině případů mě nezajímá ani jejich velikost (jednu z mála výjimek může představovat nutnost rozpoznání kvádru a krychle, kdy obě tělesa mají stejný tvar, ale liší se právě rozměry jednotlivých ploch).

Pouhým odebráním ploch ze vstupního pole by však zároveň došlo k výrazné ztrátě důležitých informací. Plochy totiž udávají, kde jednotlivé hrany leží. Zároveň předávají hranám informaci o úhlu, jejž svírají se sousedními plochami.

Pro úspěšné rozpoznávání je důležité především prostorové rozmístění hran v poli terminálů. Předpokládám sice, že vstupní pole bude převedeno na řetězec terminálních symbolů, nicméně posloupnost terminálů v tomto řetězci je dána právě tímto prostorovým uspořádání jednotlivých hran. Jinými slovy samotné plochy, i když nebudou popsány přímo gramatikou, ignorovat nemohu.

3.5.2 Problém natočených objektů

Výše navržený systém popisu objektů skládajících se pouze z hran je možné prohlásit opět za funkční. Základem by opět byla maticová gramatika, která by popisovala prostorové rozmístění hran v rámci daného tělesa. Tuto gramatiku je rovněž možné převést na bezkontextovou a sestrojit pro ni příslušný syntaktický analyzátor. Zároveň díky omezení počtu terminálů je pravděpodobné, že výsledná gramatika bude obsahovat méně nejednoznačností, tudíž bude méně složité sestrojit rozhodovací LR tabulku.

V případě hran vedoucích horizontálním a vertikálním směrem bude popis takto utvořenou gramatikou dobře čitelný. Jednotlivé dílčí hrany budou na sebe navazovat a pomocí gramatiky je snadno zredukuji do agregovaných hran.

Pokud však objekt bude obsahovat hrany vedoucí diagonálním směrem (což je případ natočených objektů), nastane problém s derivačními pravidly. Zpracováním vstupního pole terminálů po sloupcích dojde k porušení posloupnosti terminálů označujících jednu hranu mezi sebou. To samo o sobě neznamená, že by pro takto popsané objekty nešla sestrojit vhodná gramatika. Dá se ale předpokládat, že nastalá skutečnost by vyžadovala definovat velkou řadu pravidel pro jednotlivé objekty.

Tato skutečnost opět povede ke komplikacím při konstrukci LR tabulky. Stejně tak počet nejednoznačností zůstane relativně vysoký. Je tedy třeba zamyslet se, zda by nešel zjednodušit celý systém popisu objektů a kompletně tak opustit systém hran a ploch.

3.5.3 Popis tvarů objektu

Systém popisu objektů pomocí jeho hran s vynecháním ploch se opět ukázal jako funkční, ale pro mé potřeby byl stále příliš složitý. Rozpoznávání vstupních objektů na základě jejich tvaru a vzájemných úhlů představuje jistě správnou cestu k úspěchu. Je však důležité co možná nejvíce omezit počet pravidel, a tím zredukovat nejednoznačnosti v nejvyšší možné míře.

Při pohledu na samotný způsob popisu objektů, který dostávám na vstupu, tedy pole hodnot vzdáleností od středu snímání pro různé vodorovné roviny, je možné postřehnout, že již do jisté míry obsahuje informace o tvaru objektů. Zvedá-li se hodnota vzdálenosti, je jasné, že plocha na povrchu tělesa má určitý sklon, spodní hrana této plochy je dál od středu tělesa. Plocha rovněž svírá s osou snímání určitý úhel. Začne-li hodnota vzdáleností opět klesat, popřípadě se přestane měnit, dá se vytušit, že v těchto místech se nalézá klíčová hrana objektu oddělující dvě plochy od sebe.

Stojí za zamyšlení, zda by nebylo možné popsat každý objekt jako sekvenci hran a ploch pro každý úhel snímání. Tento popis může umožňovat značné zjednodušení gramatiky, protože každý objekt bude popsán osmi sloupci, ve kterých se vyskytují vždy na klíčových místech jednotlivé hrany. Odpadá tak návaznost jednotlivých dílčích hran na hrany vyskytující se v ostatních sloupcích.

Prakticky rozdělím plášť tělesa na osm částí. Každý sloupec vstupního pole popisuje osminu pláště. V prvním kroku provedu rozpoznávání pro každou část pláště. Všech osm mezivýsledků tvoří charakteristickou posloupnost, na jejímž základě dojde k identifikaci tělesa.

3.5.4 Úprava způsobu převedení pole vzdáleností na pole terminálů

Sklony jednotlivých ploch je teoreticky možné odečítat přímo ze vzdálenostních hodnot, ale tento přepočet by mohl být značně nepřesný. I pro rovné plochy se totiž může vzdálenost od středu snímání měnit.

Například plocha vedoucí kolmo na osu snímání v určité vzdálenosti bude mít pro úhel snímání 90° menší vzdálenost než pro úhly snímání 45° a 135° (konkrétně může jít o vzdálenosti 2,828 – 2 – 2,828).

Proto je vhodné zachovat systém převodu hodnot vzdáleností na body v souřadném systému a vytvoření pomyslného polygonového modelu. Při bližším pohledu na tento model zjistíme, že se skládá z osmi sloupců ploch, které odpovídají jednotlivým osminám pláště, jimiž objekt budu popisovat. Použitím stejného principu vypočtu pro jednotlivé plochy normálové vektory sloužící pro určení náklonu plochy.

Skalárním součinem opět získám úhel, který svírají dvě plochy mezi sebou. Nicméně princip výpočtu úhlů mezi plochami již bude nutné drobně modifikovat. Normálové vektory všech ploch v jednom sloupci neleží v jedné rovině. Výpočet úhlů mezi těmito plochami tak bude značně zavádějící, neboť je počítán pro rovinu, která prochází oběma normálovými vektory, nikoli pro svislou rovinu snímání v daném úhlu.

Je tedy nutné srovnat tyto normálové vektory pro plochy v jednom sloupci do roviny snímání. K tomu můžeme využít techniky popsané v kapitole 3.2.4. Pomocí 3D transformace (a pomocného vektoru) otočíme vektory do jedné roviny, kde je možné spočítat přesný úhel. O vzájemné poloze obou ploch (tedy odklonu nebo příklonu) rozhodneme rovněž totožnými metodami (opět užití dvou pomocných vektorů, resp. porovnání velikostí dvou vektorů).

Problém rotačních ploch již nadále nepředstavuje potíž. Normálový vektor se sice může lišit v závislosti na volbě bodů pro výpočet směrových vektorů, nicméně jeho transformací do roviny snímání dojde k potlačení negativního efektu.

3.5.5 Kontrola tvarů objektů a fiktivní hrany

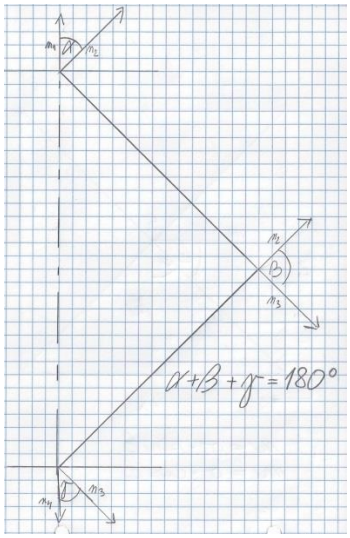
Při pohledu na trojrozměrné objekty je možné si všimnout, že vždy opíší ve svislých a vodorovných rovinách určitý úhel. Konkrétně 180° pro svislou a 360° pro vodorovnou rovinu.

Této skutečnosti se dá využít ke kontrole tvarů jednotlivých vstupních objektů. Součet všech úhlů v dané rovině musí být přibližně stejný, jako jsou výše zmíněné hodnoty (díky deformacím se mohou součty mírně lišit). Ve chvíli, kdy součet úhlů hran v dané rovině není roven předpokládané hodnotě, je patrně chybný celý vstupní soubor.

Nicméně aby vše správně fungovalo, je třeba doplnit fiktivní spodní a horní plochy do vstupního pole. Princip doplnění ploch je obdobný způsobu popsanému v kapitole 3.2.5. V tomto případě však budou plochy doplněny i v případě, že objekt žádnou horní a spodní plochu ve skutečnosti nemá (jinými slovy všechny plochy byly korektně zaznamenány a na první pohled není třeba je doplňovat).

Pomocí této fiktivní plochy je jednak možné zjistit počáteční sklon první a poslední plochy ve snímané rovině a hlavně dojde k doplnění již existujících hran v této rovině. Bez těchto fiktivních hran by se součet všech úhlů nerovnal očekávané hodnotě.

Například pro neotočený kvádr by v každé svislé popisné rovině neexistovala žádná hrana (systém by zaznamenal pouze jednu svislou plochu). Stejná situace nastane v případě dvojitého jehlanu. Zde bude existovat pouze jedna hrana s úhlem o velikosti 90°. Přidáním fiktivních rovin pak dojde k doplnění dalších hran, které jsou pro úspěch systému klíčové. Nyní již bude mít neotočený kvádr dvě hrany s úhly o velikosti 90° a dvojitý jehlan tři hrany o velikosti 45°-90°-45°. Ve všech svislých rovinách dosáhnu požadovaného součtu 180°.



Obr. 18 Zobrazení všech hran (i fiktivních) objektu v dané rovině + nastínění principu kontroly celkové velikosti všech úhlů

Pro potřeby kontroly oblouků v rovinách bude však nutné upravit systém tak, aby umožnoval ukládání velikostí úhlu k jednotlivým terminálním a nonterminálním symbolům.

3.5.6 Popis tvaru objektu ve vodorovné rovině

V předchozím textu jsem uvedl, že většina objektů může být spolehlivě odlišena na základě tvaru jednotlivých sloupců (osmin plášťů). Některé objekty však díky deformacím na vstupu mohou získat velice obdobné podoby. Takto se stanou prakticky navzájem neodlišitelné, vezmeme-li v potaz pouze tvary sloupců.

Proto je vhodné popsat tělesa ještě ve vodorovném směru. Počet rovin, pro které bude objekt popsán, může být proměnlivý. Pokud by bylo mým úkolem rozpoznávat velice podobné objekty jen s malým počtem odlišností, bylo by vhodné popsat všechny vstupní roviny.

V mém případě to však nutné není. Ke spolehlivému odlišení všech objektů, které budu rozeznávat, postačí pouze popis pro prostřední vodorovnou rovinu.

Princip popisu středové roviny bude prakticky totožný s principem popisu sloupců. Bude mě zajímat pouze rozmístění klíčových úhlů.

3.6 Definování prvků gramatiky popisující tvar

Gramatiku popisující tvar v osmi svislých rovinách a jedné vodorovné můžeme ve své podstatě opět považovat za maticovou, neboť umístěním těchto rovin vedle sebe dostaneme opět dvojrozměrné vstupní pole terminálů.

Nicméně znovu je možné využít vlastností maticové gramatiky a převést ji na bezkontextovou stejným způsobem jaký byl popsán v kapitole 3.4.1. Napřed tedy zpracuji samostatně jednotlivé roviny popisu a nakonec celý objekt. Proto budu tuto gramatiku již od začátku definovat jako bezkontextovou. Zabývat se dalšími maticovými pravidly již nemá v této fázi návrhu systému smysl.

3.6.1 Přiřazení vhodných symbolů plochám a hranám

Při přiřazování terminálů plochám a hranám využiji poznatků z předešlých kapitol. Jednotlivé plochy je možné opět zcela ignorovat. Pro rozpoznání objektů nejsou důležité, neboť se děje na základě tvaru objektů určených úhly jednotlivých hran.

Dokonce ani samotná pozice hran není příliš důležitá. V mém případě si mohu dovolit vypustit informace, mezi kterými plochami hrana leží. Dostačující je samotná posloupnost po sobě jdoucích úhlů.

Pro popis hran potom lze využít stejného systému, jako byl použit v kapitole 3.3.2. Při určení tvaru objektů totiž můžeme využít stejných vlastností. Stanovením vhodných intervalů lze omezit velké množství deformací, což povede k jednodušší gramatice.

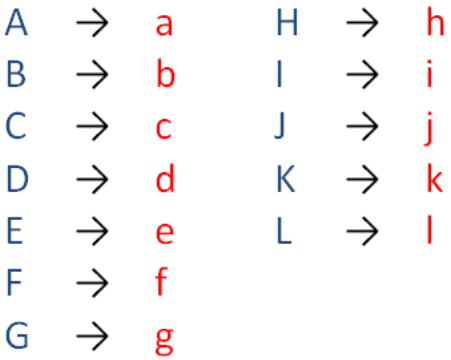
Jednotlivé hrany ve vodorovných rovinách mohou být popsané stejným způsobem jako hrany pro svislé roviny. V obou dvou případech normálové vektory svírají mezi sebou určitý úhel. Zjednodušeně je možné prohlásit, že vodorovnou rovinu lze pokládat za další svislou rovinu a stejně tak s ní pracovat.

Nonterminály je opět možné rozdělit na dvě skupiny. První skupina derivuje samotné nonterminály, druhá pak derivuje vhodně zvolené části objektů a počáteční symbol pak objekt samotný.

3.6.2 Stanovení derivačních pravidel

Derivační pravidla jsou nejdůležitější součástí gramatiky. Pomocí nich jsou popsány jednotlivé objekty, které by měl syntaktický analyzátor zpracovávající tuto gramatiku rozeznat. Rozpoznávání se děje na základě sestavení derivačního stromu.

Formát derivačních pravidel gramatiky bude vycházet z Chomského normální formy. Ta však nebude v plné míře dodržena. Stále bude platit, že jednotlivé terminály budou převáděny na nonterminály pomocí samostatných pravidel. Nedodržím však princip, kdy na pravé straně musí být nanejvýš dva nonterminály.



Obr. 19 Pravidla derivující terminály z nonterminálních symbolů

Použitím vhodných intervalů pro jednotlivé úhly hran bylo možné odstranit většinu deformací. Stále je však nutné uvažovat o gramatice jako o nedeterministické. Zbylá zkreslení je totiž nutné popsat pomocí doplňujících pravidel, která derivují ze stejného nonterminálu více možných kombinací. Tato skutečnost se týká především symbolů B, D, I a K.

Obr. 20 Dílčí derivační pravidla

Symboly *#* a *$* pak představují oblouky. Do těchto oblouků jsou redukovány posloupnosti malých úhlů (do 40°). To umožňuje následně zjednodušit popis jednotlivých svislých a vodorovných rovin.

1	→	M	10	→	GJG
2	→	N	11	→	GLL
3	→	O	12	→	CDC
4	→	D#C	13	→	G$#
5	→	C#C#C	14	→	GI$I#
6	→	C#D	15	→	D$BD
7	→	#	16	→	D$#BD
8	→	FKF	17	→	D$CD
9	→	GLG	18	→	D$#CD

Obr. 21 Pravidla pro popis svislých rovin (pozn.: číslice jsou brány jako jeden symbol)

Každý nonterminál označený číslicí pak popisuje jednu samostatnou rovinu. Zde je nutné zmínit, že je nezbytně nutné, aby byly takto popsány všechny možné roviny pro všechny vstupní objekty. Pokud by tomu tak nebylo, docházelo by k chybným detekcím.

Na Obr. 21 se nacházejí pravidla pro popis svislých rovin. Nalezneme zde oba popisy neotočených a otočených objektů, které splňují všechna omezení definovaná v kapitole 3.1.3. Na Obr. 22 je množina pravidel doplněna o popis vodorovných rovin.

20	→	PP	23	→	TT
21	→	QQ	24	→	MM
22	→	RR	25	→	NN

Obr. 22 Pravidla pro popis vodorovné roviny (pozn.: číslice jsou brány jako jeden symbol)

Po detekci všech osmi svislých a jedné vodorovné roviny je možné přistoupit k finální fázi rozpoznávání. Na zásobníku by se mělo nacházet devět nonterminálů, přičemž na vstupu by se měl nacházet pouze koncový symbol.

3.7 Syntaktický analyzátor pro gramatiku popisující tvar

I když byly nejednoznačnosti v gramatice omezeny na minimum, stále se jedná o gramatiku nedeterministickou. To lze doložit na pravidlech B → XX a X→XX. Ve stavu XX existují dvě možné redukce, navíc ve chvíli zpracování není jasné, které pravidlo je správné.

Z tohoto důvodu nelze využít top-down automatů (nakonec se nejedná ani o LL gramatiku). Automaty s přístupem seshora-dolů kromě jiného nejsou vhodné k detekci deformovaných objektů, protože detekci chyby odkládají na nejzazší možný okamžik. Silnější automaty s bottom-up přístupem (např. LR parser) sice již

dokáží mnou navrženou gramatiku zpracovat, ani ty si však neporadí s deformacemi.

Pro práci s nejednoznačnými gramatikami je nutné použít ještě silnější prostředky, například modifikované bottom-up parsery založené na LR tabulce. Ty již dokáží odhalit chyby v momentě jejich načtení, obsahují mechanismy, které se dokáží s nejednoznačnými situacemi vyrovnat, navíc ke své práci stále využívají rozhodovací tabulky a nemusí tak prohledávat celý stavový prostor. Mezi nejznámější zástupce patří Tomita parser (4) a LRE(k) parser (9) (též Fast Early parser).

Tomita parser vytváří pro každou z možností, kterými se může vydat, nové vlákno. Tato vlákna jsou zpracovávána nezávisle na sobě a až po skončení analýzy jsou vyhodnocena. Jako správné je pak považováno vlákno s nejmenším počtem odchylek.

Naopak LRE(k) parser, který v sobě slučuje postupy Earlyho a LR parseru, zpracovává nejednoznačnosti postupně, díky čemuž je vhodnější, pokud je nutné implementovat překladač. Při zpracovávání více vláken současně se obtížně provádí sémantické operace, neboť v době analýzy nelze určit správné vlákno. Tuto nevýhodu LRE(k) parser odstraňuje.

V mém případě se však nemusím sémantickými operacemi zabývat, neboť k detekci slouží samotné sestavení derivačního stromu. Proto je vhodnější použít Tomita parser, který je navíc jednodušší na implementaci.

3.7.1 Konstrukce LR tabulky

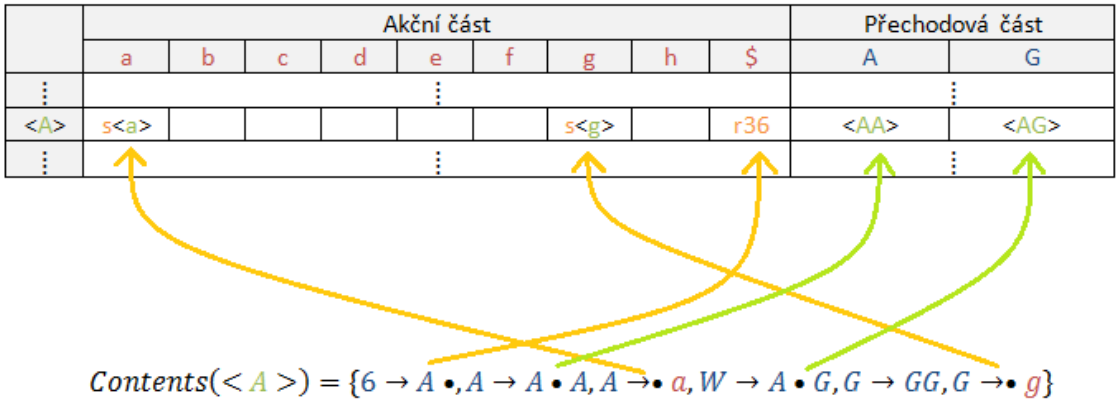
Jak již bylo nastíněno výše, Tomita parser využívá pro svoje rozhodování LR tabulku. V tabulce jsou obsaženy informace o činnostech, které lze pro daný vstup vykonat, a o následných stavech, které následují za současným stavem. Zároveň usnadňuje automatu zotavení se z chyb vzniklých v důsledku deformací na vstupu. Ve chvílích nepředpokládaného vstupu pro aktuální stav, nabízí LR tabulka možnosti, kterými se lze dále ubírat. Ve většině případů je tak možné pokračovat v syntaktické analýze bez jejího přerušení.

Než bude možné přikročit ke konstrukci tabulky, je nutné doplnit gramatiku o několik nových množin a prvků stávajících množin.

- Startovací pravidlo S‘→S má v podstatě pomocný charakter, ukončuje samotnou analýzu úspěšným sestavením derivačního stromu.
- Položka udává aktuální pozici na vstupu při zpracování pravidla. Obvykle je značena tečkou.
- Množina uzávěrů položky pak obsahuje samotnou položku a všechny ostatní položky, které se z ní dají derivovat.
- Množina prefixů pravých stran je důležitá především pro stanovení všech možných stavů LR tabulky, kdy každý prvek množiny odpovídá právě jednomu stavu.
- A nakonec nejdůležitější množiny *Contents(x)* pro každý stav *x*. Pomocí této množiny jsou v akční části stanoveny jednotlivé akce pro každý možný vstup.

Pro přechodovou část tabulky také udává informace o následném stavu. Jinak řečeno, bez této množiny by nebylo možné zkonstruovat LR tabulku.

Pro konstrukci LR tabulky jsou potřebné hlavně množina prefixů a množiny *Contents(x)*. Zbylé slouží především k definování prvků jmenovaných množin. Tabulku lze konstruovat jako statické dvourozměrné pole (tedy paměť s přímým přístupem). To zajistí rychlé vyhledávání informací pro každý stav.



Obr. 23 Princip tvorby obsahu LR tabulky
Zdroj: Návrh systému pro aplikaci deformačních gramatik, 2014 (16)

3.7.2 Problémy pramenící z použití LR tabulky

Nevýhodu LR parserů představuje náročná konstrukce samotné rozhodovací tabulky. Ta může být značně rozsáhlá v závislosti na počtu a velikosti derivačních pravidel. Pro podobné objekty s malými rozdíly (to může být případ i tvaru hran) pak může syntaktický analyzátor často procházet stejnými stavy v různých fázích analýzy. Tato skutečnost vede k nutnosti popsat v těchto stavech všechny možné eventuality.

Například pokud by existovaly pravidla K → ABC a L → ABD. LR tabulka by pak obsahovala stavy <A>, <AB>, <ABC> a <ABC>. Ve stavu <AB> pak musí existovat oba možné přechody, do stavu <ABC> nebo <ABD>. Těmito stavy bude automat procházet vždy, kdy zaznamená danou kombinaci nonterminálů. Nicméně určité kombinace nemusí v určitých fázích dávat smysl. Pokud se nonterminál L může vyskytovat jen v první třetině popisu objektu, pak uvažovat o přechodu do stavu <ABD> nemá v pokročilejších fázích analýzy smysl. Přesto pokud se na daném místě vyskytne chyba, bude syntaktický analyzátor prohledávat i tento už nesmyslný stav.

Díky tomu dochází ke snížení efektivity samotného parseru. Velký počet možností pro daný stav tak znamená také nutnost prozkoumat tyto možnosti při zotavování z chyb. Dojde k vytvoření velkého počtu vláken, kde většina z nich vznikla z chybné predikce automatu. Přesto se jimi musí analyzátor zabývat, což prodlužuje jeho činnost.

Tato chybná vlákna pak často generují další chyby. Jelikož jsou založena na nesprávných předpokladech, stav automatu a zásobníku se odchýlí od předpokládané konfigurace. Díky tomu je nový vstupní terminál opět označen za chybný, což vede k sérii opravných kroků. Tento postup je ukončen až ve chvíli, kdy vlákna dosáhnou maximální tolerované chybovosti a jsou zahozena. Předtím si však vyžádají velké množství výpočetního času.

Další nevýhodu LR parseru představuje jistá nepružnost celého systému. Ve chvíli, kdy se automat přesune z počátečního stavu na druhý, předpokládá, že již zpracovaný nonterminál je správný. Syntaktický analyzátor tak očekává pouze vstupy vyplývající ze současného stavu zásobníku. V případě, že stav zásobníku je chybný, bude chybný i další postup parseru, neboť je opět tento postup založen na nesprávných předpokladech.

Pro deformační gramatiky je však charakteristické, že na začátku je přípustné velké množství stavů. Automat jednoduše neví, které terminály jsou způsobené deformacemi a které představují skutečný stav. Musí tedy všechny rozpracovat.

Situace znovu povede k vytváření velkého množství vláken při možných chybných vstupech, kde většina těchto vláken bude představovat chybné předpoklady, jež budou prodlužovat celkovou dobu analýzy.

3.7.3 Nezávislé zpracování rovin

Oběma problémům lze předejít osamostatněním jednotlivých fází syntaktické analýzy. Při podrobnějším pohledu na objekty si lze uvědomit, že se většina z nich skládá ze stejných osmin pláště (tedy sloupců se stejným tvarováním). Tyto stejně tvarované sloupce se pak v jednotlivých objektech vyskytují ve velice proměnlivém pořadí. Může být konstatováno, že po zpracování určité roviny automat nemůže spolehlivě prohlásit, který tvar má následovat v další rovině.

Zpracování každé roviny v podstatě můžeme pokládat za samostatnou úlohu. Detekováním určitého typu části pláště dojde k přechodu do finálního stavu a ukončení dané části analýzy. Pozitivním efektem bude velikost výsledné LR tabulky. Což umožní její jednodušší implementaci a efektivitu rozhodování pro případné chyby.

Zpracováním jednotlivých popisných rovin samostatně odpadnou největší problémy spojené s LR tabulkou. Automat nebude procházet opakovaně stejnými stavy, sníží se tak množství eventualit, které bude muset LR tabulka obsahovat. Následná náprava chyb se tak stane efektivnější, neboť možná pokračování budou vycházet z aktuální pozice vstupu, nikoli ze všech možných vstupů pro daný stav. Tím se omezí tvorba chybných vláken a zrychlí se zpracování vstupu.

Další příznivou vlastností tohoto způsobu řešení je jistá ztráta „paměti“ automatu. Po každé zpracované rovině se automat může přesunout zpět do výchozího stavu. Další průběh rozpoznávání tedy nevychází ze stavu předešlého. Opět tím mírně snížíme množství stavů. Odpadne nutnost pokrýt možnost, že se automat nachází ve špatném stavu vlivem chybné detekce předešlé části objektu a oprava chyb bude efektivnější.

Drobnou nevýhodu naopak představuje nebezpeční chybné detekce jedné hrany. Stejně tak je nutné držet v paměti dílčí výsledky pro jednotlivé roviny.

3.7.4 Rozdělení syntaktického analyzátoru na dvě části

Řešení potíží vzniklých ze samostatného zpracování hran je pak možné spatřit v rozdělení syntaktického analyzátoru na dva samostatné celky, které budou na sobě navzájem nezávislé.

Výsledek prvního automatu v podobě detekce dílčího tvaru v určité rovině bude zároveň představovat vstup druhého automatu. Než druhý automat zahájí svoji činnost, vyčká, dokud nedojde k rozpoznání tvarů pro všechny vstupní roviny. Následně provede syntaktickou analýzu pro devět vstupních nonterminálů, ze kterých utvoří finální derivační strom, na jehož základě rozpozná daný objekt.

Ač se bude jednat o dva automaty, není třeba vytvářet novou gramatiku, neboť mohu využít již definovaných derivačních pravidel. Automat pracuje s terminály a nonterminály jako s čísly, proto je při jeho práci možné nahradit oba druhy symbolů. Po formální stránce není druhá gramatika úplná, nicméně vše bude funkční, o což jde především.

Tímto způsobem se elegantně vyřeší i případný problém s chybnou detekcí hrany. Budu-li pokládat gramatiku druhého automatu opět za deformační a vyberu pro její zpracování znovu Tomita parser, případné nepřesně rozpoznané hrany budou opraveny pomocí stejných prostředků, jako tomu je pro standardní vstup terminálů.

V některých případech, pokud výsledky dílčích analýz neobsahují chybné detekce, je možné využít i jednodušších parseru, pro jednoznačné gramatiky, které obecně pracují o něco rychleji.

V rámci implementace pak mohu využít objektového paradigmatu, kdy pro obě samostatné úlohy bude použit stejný obecný automat, dojde jen ke změně částí souvisejících s danou gramatikou. Největší změnu pak představuje nutnost přesunout centrum řízení rozpoznávání ze syntaktického analyzátoru na řídicí systém, jenž bude parser obalovat.

3.7.5 Systém podmíněných pravidel

Vzhledem k tomu, že gramatika popisující tvar v daných rovinách je stále deformační, musí v ní existovat i pravidla, která tyto deformace popisují. To často vede k nejednoznačnostem. S těmito nejednoznačnostmi se dovede Tomita parser poměrně snadno vyrovnat, nicméně zpracování většího počtu vláken může zdržovat celý mechanismus rozpoznávání.

Při hlubší analýze je možné rozpoznat, že některá deformační pravidla nedávají za určitých situací smysl. Například může vzniknout deformace, kdy pravý úhel je rozdělen na dva menší úhly a jeden úhel o velikosti 45°. Tato situace je popsána pomocí pravidla $D \rightarrow XBX$.

Při pohledu na stanovené intervaly v kapitole 3.3.2 je možné zjistit, že pod nonterminálem X se mohou skrývat různé úhly do 40°. Například pokud by první

nonterminál X představoval úhel o velikosti 35° a druhý úhel o velikosti 30°, potom by výsledný „pravý“ úhel měl velikost 35+45+30 = 110°.

Na první pohled je jasné, že vlákno zpracovávající toto deformační pravidlo je chybné a založené na mylném předpokladu. Maximální velikost pro pravý úhel je stanovena na 95°, což je o 15° méně, než je popsaný případ.

Mohu tedy stanovit, že dané vlákno bude pokračovat pouze za předpokladu, že součet všech zúčastněných úhlů spadá do předpokládané velikosti. To celé povede k omezení chybných vláken a zrychlení analýzy.

Pro funkci této části systému je navíc možné využít již navržené části systému, která k jednotlivým terminálům a nonterminálům ukládá velikosti příslušných úhlů.

4 Implementace systému pro rozpoznávání trojrozměrných objektů

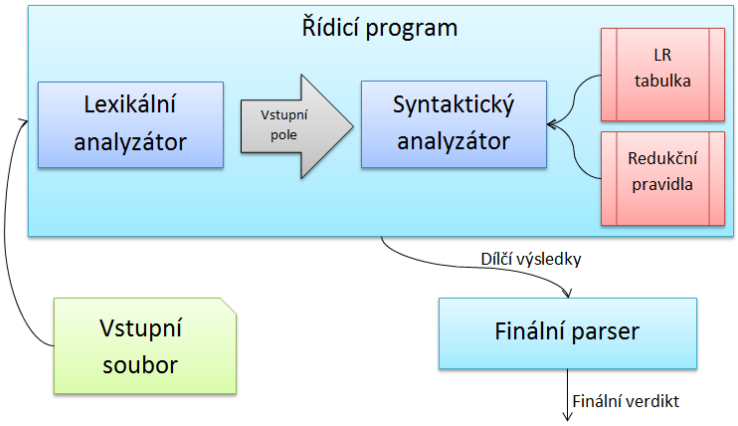
V následující kapitole bude popsána implementace samotného systému, jenž byl navržen v předešlé kapitole. Půjde o variantu Tomita parseru, který bude postupně zpracovávat jednotlivé tvary objektu ze všech snímaných rovin. Nakonec provede finální syntaktickou analýzu všech dílčích výsledků a určí, o jaký objekt se jedná.

Systém je implementován v jazyce C++ jako objektová aplikace ve vývojovém prostředí NetBeans od společnosti Oracle. Díky využití objektové architektury bylo možné začlenit do mého systému již vytvořený syntaktický analyzátor. V rámci mé práce jsem však musel vytvořit ostatní systémy spolupracující s parsrem a hlavně definovat novou gramatiku popisující 3D objekty.

4.1 Jádro syntaktického analyzátoru pro rozpoznání 3D objektů

Jádro systému tvoří samotná implementace Tomita parseru, který jsem prezentoval v rámci mé bakalářské práce (16). Pro potřeby aktuálního systému však bylo nutné učinit řadu úprav. Tou největší byla samotná změna konceptu fungování celého analyzátoru.

V předešlé práci systém fungoval na podobném principu, jaký lze pozorovat u syntaxí řízených překladů. Parser řídil všechny ostatní systémy (načítání nových symbolů, zásobník, seznam vláken, rozhodovací tabulky, …) a jejich metody volal pouze ve chvílích, kdy od těchto systémů vyžadoval určité činnosti.



Obr. 24 Schéma celého systému

Vzhledem k tomu, že současný systém musí zpracovat více úloh za sebou, bylo nutné tento koncept opustit, respektive obalit obslužnými metodami. Lexikální analyzátor nyní funguje nezávisle. Načte vstupní soubor se všemi vzdálenostními hodnotami a vytvoří pro každou snímanou rovinu vstupní pole terminálů.

Teprve poté je zavolán syntaktický analyzátor, který zpracovává jednotlivá vstupní pole. Aby úpravy nemusely být příliš rozsáhlé, je nutné zachovat onu představu, že parser stále řídí celou analýzu. Obslužné metody tedy do určité míry „emulují“ předchozí princip. Syntaktický analyzátor tak stále využívá stejných metod a signálů jako v předchozí práci, ale už neřídí samotný průběh rozpoznávání.

4.1.1 Syntaktický analyzátor

V následujícím textu krátce zrekapituluji nejdůležitější vlastnosti již implementovaného syntaktického analyzátoru. Jedná se o implementaci Tomita parseru, které dokáže zpracovat nejednoznačnou gramatiku. Obsahuje mechanismy, jež si dokáží poradit s konflikty mezi více REDUCE pravidly, tak i konflikty SHIFT/REDUCE, kdy není jasné, zda uplatnit pravidlo, nebo načítat další tokeny na zásobník.

Řešení konfliktů odpovídá principu Tomita parseru, pro každou možnost je vytvořeno separátní vlákno, jež je zpracováváno autonomně. To si vyžádalo zavedení systému signálů, pomocí nichž vlákna udávají, ve kterém stavu se aktuálně nachází (např. že vlákno čeká na načtení nového tokenu, obsahuje chybu a má být smazáno nebo dosáhlo konečného stavu).

Analýza končí, pokud jsou všechna vlákna v důsledku chybovosti smazána, nebo dosáhla konečného stavu. V případě, že se v konečném stavu po skončení analýzy nacházelo více vláken, systém vybral za správné to s nejmenší chybou.

Posledním důležitým celkem syntaktického analyzátoru byly metody snažící se urychlit zotavení z chyb. Za normálních okolností, pokud systém narazil na chybu, snažil se z ní dostat obdobným principem, jako při konfliktu. Vytvořil vlákno pro všechny možnosti pokračování.

Ve většině případů je chyba způsobena chybějícím, nebo naopak přebývajícím tokenem. Metody se tak snažili najít správný stav odstraněním aktuálního tokenu, nebo naopak prozkoumáním dalších možných stavů pro aktuální token. Až v případě selhání se automat uchýlil ke standartnímu procesu zotavení. Tento systém byl však volitelný a automat fungoval i bez něj (v této práci je využíván).

4.1.2 Úpravy výchozího automatu

Vzhledem k tomu, že syntaktický analyzátor byl konstruován jako obecný, nebylo prakticky třeba provádět výrazné úpravy vzhledem ke změně zpracovávané gramatiky (to pochopitelně neplatí pro LR tabulku a implementaci redukčních pravidel, která jsou naopak striktně vázána na aktuální gramatiku). Ta je opět bezkontextová, tudíž je možné ji zpracovat pomocí stejných principů.

Jedinou výjimku představuje systém podmíněných pravidel. Všechna vlákna bylo třeba rozšířit o informaci o velikosti úhlu, jež představují. Rovněž byl dodán mechanismus signalizující, že aktuální vlákno pro dané pravidlo nesplňuje podmínku a je třeba jej smazat. V případě nesplnění podmínky obdrží vlákno signál ERR_RED, tedy že na zásobníku nejsou dostupné správné tokeny pro aplikaci daného pravidla.

Zásadní změny pak souvisí s odstraněním principu syntaxí řízeného překladu. Lexikální analyzátor byl vyčleněn ze struktur automatu a získal při své činnosti určitou nezávislost. Parseru je pak do konstruktoru z obalujícího systému předáván pouze odkaz na scanner, který již ukončil svoji činnost. Syntaktický analyzátor tak už nemůže do jeho práce zasahovat, natož jej řídit. Metody zjišťující následné symboly, zbývající počet tokenů ve vstupním poli atd. však zůstaly zachovány.

Další neméně důležité úpravy si vyžádala potřeba zpracovat několik rovin v rychlém sledu krátce za sebou. Před započetím analýzy následující roviny je třeba automat resetovat a vrátit do výchozího stavu.

Nakonec jsem upravil i systém počítání chyb. V předchozí verzi pro 2D objekty mohlo dojít k situaci, kdy určité tokeny přebývaly nebo naopak chyběly. Vzhledem k rozdílům ve způsobu vzniku možné deformace, však tyto situace v současném systému nenastávají. Jeden konkrétní úhel může být rozbit na více menších úhlů, nicméně počet tokenů ve vstupním poli je vždy konstantní.

Chyba se nyní počítá pouze ve chvíli, kdy je aplikováno deformační pravidlo, tedy došlo k odhadu na základě výše popsané situace. V případě, že je tento odhad chybný, má toto vlákno vždy vyšší chybovost a samo se diskvalifikuje při rozhodnutí o tvaru roviny.

4.1.3 Lexikální analyzátor

Lexikální analyzátor je druhou nejdůležitější součástí celého systému pro rozpoznávání trojrozměrných objektů. Při zahájení analýzy načte pomocí speciální třídy vstupní textový soubor obsahující jednotlivé hodnoty vzdáleností. Jsou zkontrolovány rozměry tohoto pole a nakonec je celé převedeno na body v trojrozměrném souřadném systému pomocí goniometrických funkcí. Dále jsou z nově vypočtených bodů vytvořeny pomyslné plochy. Těm jsou dopočítány směrové vektory a následně i vektor normálový.

Ve druhé fázi lexikální analyzátor zpracovává jednotlivé roviny snímání a vypočítává úhel mezi normálovými vektory. Při této příležitosti jsou občas aplikovány 3D transformace, jež srovnávají vektory do jedné roviny tak, aby vypočtený úhel nebyl zkreslený.

Každý úhel je následně převeden na příslušný terminál pomocí jednoduchého switche. Terminál je pak zařazen na konec vstupního pole náležícího k dané rovině. Pro každou rovinu vzniká samostatné vstupní pole, na jehož konci je symbol označující jeho konec.

Drobnou komplikaci představuje poslední vodorovná rovina, kdy se pořadí tokenů (úhlů) může lišit v závislosti na natočení objektu kolem osy *y*. Byla potřeba určit množinu počátečních terminálů a v případě, že se takový symbol na začátku pole neobjevuje, je nutné pole otočit (první symbol přesunout na konec pole a ostatní přemístit o pozici dopředu).

Pro potřeby syntaktické analýzy lexikální analyzátor obdrží od obalujících metod aktuální rovinu, ze které má vracet tokeny. Ty parser načítá podle původních metod, nicméně nyní již scanner nepracuje s otevřeným souborem (nebo polem), který postupně načítá a zkracuje. Namísto toho udržuje informaci o aktuální pozici

v právě zpracovávaném poli. Celý princip je nedestruktivní, jinými slovy každou rovinu lze zpracovat znovu.

4.1.4 Obalující řídicí systém

Jak již bylo zmíněno výše, celý proces rozpoznávání je řízen speciálním obalujícím systémem. Ten ve správném pořadí volá jednotlivé dílčí subsystémy a celkově odpovídá za výsledek analýzy.

V první řadě obsluhující systém spustí lexikální analyzátor. Ten zpracuje vstupní pole a připraví jednotlivé vstupy. Pokud nedošlo k chybě, dochází postupně k syntaktické analýze pro všechny vstupní roviny. Jednotlivé výsledky pro dílčí roviny jsou ukládány do pomocného pole tvarů.

Toto pole je využito při závěrečné analýze, kdy je spuštěn finální parser, který posoudí jednotlivé mezivýsledky a na jejich základě určí trojrozměrný objekt, jenž je výsledkem celkové analýzy.

Protože se pořadí svislých ploch může opět měnit v závislosti na natočení vstupního objektu, je třeba přetočit pole mezivýsledků do výchozího stavu. To je prováděno obdobným způsobem, jako v případě lexikálního analyzátoru.

4.2 Další podpůrné systémy

Mezi jednotlivé podpůrné systémy patří především finální parser, který posuzuje dílčí mezivýsledky a rozhoduje o konečném verdiktu celého systému, třída načítající vstupní pole a části sloužící hlavnímu syntaktickému analyzátoru při jeho činnosti (LR tabulka, redukční funkce).

4.2.1 Finální parser

Pro konečné posouzení dílčích rovin a rozhodnutí o analyzovaném objektu jsem se rozhodl sestrojit samostatný parser. Tomita parser je postaven tak, aby dokázal určit správný objekt i přes jeho deformovaný popis. Automat posuzuje vlákna podle počtu chyb, které během analýzy musel opravit. Vlákno s nejmenší chybou je pokládáno za správné.

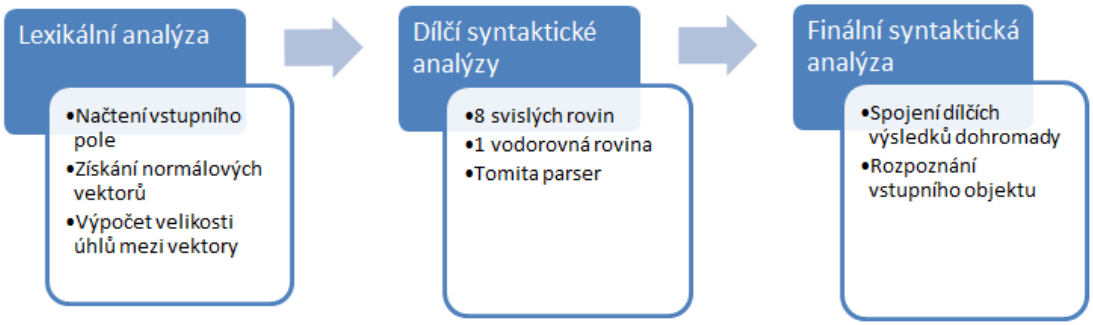
Při analýze velmi podobných tvarů však může nastat situace, kdy více vláken má stejnou míru chybovosti. V takovém případě musí parser rozhodnout na základě jiných faktorů.

Tyto situace však v mém systému nenastávají. Díky zjednodušení gramatiky a samostatnému zpracování rovin nedochází prakticky k chybným detekcím z důvodu prozkoumávání velkého počtu eventualit. Navíc, pokud už dojde k vytvoření dílčích vláken, následující průběh analýzy se pro obě možnosti natolik liší, že chybná předpověď je prakticky ihned zahozena.

Díky tomu mohu s jistotou prohlásit, že detekované výsledné hrany jsou správné. V takovém případě není nutné provádět finální analýzu složitějším Tomita parsrem, protože finální pole již neobsahují deformace.

Systém konečného posouzení vychází z rekurzivního sestupu, kdy na základě aktuálního vstupu jsou volány konkrétní metody. Celý princip je však možné radikálně zjednodušit vzhledem k malému počtu všech možných vstupních tokenů.

Další výhodou oproti hlavnímu syntaktickému analyzátoru je menší časová náročnost, neboť se celý systém skládá pouze z přepínačů a podmínek.



Obr. 25 Souhrn jednotlivých činností parseru

4.2.2 Ostatní subsystémy

Pro potřeby zajištění podpůrných činností jsem implementoval další třídy se specifickými úzce definovanými metodami.

LR tabulka je konstruována jako velké dvojrozměrné pole s přímým přístupem. Díky tomu je zajištěno rychlé vyhledávání následných stavů a prohledávání všech eventualit. Plnění pole všemi hodnotami probíhá v rámci konstruktoru.

Systém implementující redukční pravidla je konstruován jako velký přepínač. Od syntaktického analyzátoru obdrží číslo pravidla, na základě něj provede příslušné operace. Vždy zkontroluje tokeny na zásobníku, a pokud odpovídají pravidlu, nahradí je novým nonterminálem. Zároveň určí další stav automatu. Nově rovněž posuzuje velikost úhlů při podmíněných pravidlech.

V samostatném datovém souboru jsou pak drženy všechny důležité konstanty, se kterými systém pracuje. Jsou zde definovány signály parseru, kódy chybových odpovědí nebo hodnoty tokenů…

Mezi podpůrné systémy je možné zařadit i systém načítání vstupního souboru nebo zásobníky a seznamy, jež využívá parser ke své činnosti.

5 Zhodnocení výsledků navrženého systému

Implementovaný systém jsem testoval na své domácí počítačové sestavě – CPU Intel i5 3,9 GHz, 16 GB RAM a SSD disk. Program jsem spouštěl přímo ve vývojovém prostředí, které umožňuje změřit dobu trvání běhu programu.

Důležitým aspektem kromě správného rozpoznání objektu na vstupu je i rychlost celého rozpoznávání. V průmyslových zařízeních jsou kladeny na podobné systémy přísné nároky ohledně časové náročnosti analýzy. Čím více objektů dokáže systém posoudit, tím rychleji mohou výrobní linky pracovat.

Většinu testovaných objektů (představených v kapitole 3.1.1) jsem analyzoval pro několik možných natočení a velikostí, které splňovaly výchozí podmínky. Šlo mi o potvrzení, že mnou navržený koncept je funkční a reálně použitelný. V různých úhlech totiž dochází k odlišné míře deformací, proto se úhly v jednotlivých rovinách budou odlišovat. Zároveň jsem tímto způsobem testoval lexikální analyzátor, který musí srovnat normálové vektory pro různé směry vždy do správných rovin.

5.1 Testování rychlosti a správnosti analýzy

Každý test jsem provedl 10x, abych vyloučil případný vliv skryté práce operačního systému, který může, pokud je vytížen, prodloužit dobu práce programu.

Všechny testovací objety jsou rozpoznávány korektně. U zvětšeného otočeného kvádru sice dochází ke skutečnosti, že více vláken pro dílčí roviny skončí v závěrečném stavu, nicméně systém určí správný výsledek na základě chybovosti, respektive počtu užití deformačních pravidel.

V Tab. 5 je možné spatřit průměrnou dobu v milisekundách potřebnou pro správné rozpoznání objektů. V závorce je pro srovnání uveden nejvyšší naměřený čas.

5.1.1 Testované objekty

Vzhledem k tomu, že systém slouží pouze jako potvrzení konceptu, není testovací množina objektů příliš rozsáhlá. Nicméně objekty a jejich natočení jsem volil tak, abych reprodukoval všechny předpokládané situace, jež mohou v reálném světě nastat.

Různé varianty kvádrů mají za cíl prozkoumat správné vyrovnání normálových vektoru a výpočet úhlů mezi rovinami. Zvětšením velikosti objektů je posuzována schopnost poradit si s nově vzniklými deformacemi, kdy je jeden úhel rozdělen na více dílčích úhlů. Prohnutý válec má za úkol otestovat schopnost systému posoudit i konkávní tvary rovin. Další varianty objektů testují nezávislost systému na pořadí svislých rovin apod. Kompletní seznam objektů a testované vlastnosti najdete v příloze B.

Tab. 5 Čas potřebný k analýze a rozpoznání objektů pro vybrané vstupy (v závorce je uveden nejvyšší dosažený čas)

Vstupní objekt	Doba trvání analýzy (ms)
váza, tvar 1	35 (50)
váza, tvar 2	30 (38)
váza, tvar 2, zmenšená	30 (42)
Obrácený osmistěn	32 (49)
Obrácený osmistěn, otočen o 45 stupňů kolem osy y	33 (41)
Obrácený osmistěn, otočen o 90 stupňů kolem osy z	30 (40)
Válec, s prohnutým pláštěm	28 (50)
Kvádr, otočen o 45 stupňů kolem osy z	27 (29)
Kvádr, otočen o 45 stupňů kolem osy z, zvětšený	28 (35)
Osmistěn, otočen o 45 stupňů kolem osy x	32 (41)
Osmistěn, otočen o 90 stupňů kolem osy z	25 (35)
Koule	30 (41)

5.1.2 Zhodnocení výsledků

Z výsledků je patrné, že analýza objektů probíhá velice rychle. Ve většině případů jsou objekty rozpoznány do 35 ms. Přitom je třeba si uvědomit, že pro každý objekt je třeba vykonat deset syntaktických analýz za sebou (osm svislých rovin, jedna vodorovná + závěrečná analýza, kdy jsou spojeny výsledky dílčích analýz dohromady).

K vysoké rychlosti přispívají hlavně dva faktory. Zaprvé vysoká míra zjednodušení navržené gramatiky ve smyslu malého počtu nejednoznačností. Za druhé nezávislé zpracování hran.

V mé bakalářské práci (16), kde jsem užil stejný systém, dobu potřebnou k analýze prodlužovala nejvíce skutečnost, že systém vytvořil značné množství alternativních vláken, a proto musel procházet velké množství stavů. Potýkal jsem se vlastně s problémy popsanými v kapitole 3.7.2.

Zjednodušením gramatiky jsem eliminoval většinu nejednoznačností. Obecně platí, že při každé nejednoznačnosti vznikne jedno korektní vlákno. Další vzniklá vlákna jsou ale založená na chybném předpokladu. Minimalizováním nedeterminismů tedy dojde k redukci tvorby špatných odhadů.

Nezávislé zpracování rovin má rovněž nezanedbatelný pozitivní efekt. Došlo totiž ke snížení počtu přechodů mezi stavy. Díky tomu je LR tabulka daleko přímočařejší. Při zotavení z chyb jsou navrženy pouze stavy, které mají v dané fázi smysl. Navíc, díky podmínkám v pravidlech, jsou chybné odhady často rychle odhaleny a související vlákna zahozena.

5.2 Limity systému a možná zlepšení

Nejvíce limitující pro samostatný systém je především vytvořená gramatika. Aby vše korektně fungovalo, je nutné každý trojrozměrný objekt popsat sadou deformačních pravidel. V opačném případě je sice možné, že systém vstupní objekt zpracuje, nicméně určí jej za jiný, značně deformovaný předmět. Daleko pravděpodobnější je druhá možnost, tedy že parser nerozezná objekt vůbec.

Přidání pravidel do gramatiky však nutně vede k potřebě přebudovat LR tabulku. Drobné zásahy je možné provést do již existující varianty, při razantnější změně bude ale nutné navrhnout tabulku zcela znovu. Návrh tohoto typu rozhodovací tabulky je poměrně pracný. Navíc kromě ní je třeba přidat i implementaci redukčních pravidel. Částečně může pomoci generátor LR tabulky, který odvede určitou část práce.

Systémy pracující s LR tabulkou tak nejsou vhodné pro situace, kdy se předpokládá, že množina objektů bude dále růst. Ostatně tuto vlastnost je možné pozorovat i v mé konkrétní implementaci. Syntaktický analyzátor korektně a v rychlém čase rozpoznal všechny objekty. Avšak u objektů, jež nejsou popsány pravidly, selže.

Řešením může být vytvoření systému, který dokáže vygenerovat všechny doprovodné systémy automaticky (tedy nejen LR tabulku). Další možností je vymyslet určitý způsob zobecnění. Syntaktický analyzátor vrátí určitý výsledek i pro neznámé objekty. Rozpoznání objektu pak proběhne až na základě tohoto výstupu. Celý proces by však znamenal zásadní přepracování současného konceptu.

Vzhledem k nezávislému posuzování jednotlivých rovin stojí za zvážení, zda nenavrhnout aplikaci jako více procesovou. V současné době jeden proces postupně zpracovává jednotlivé roviny postupně jeden po druhém. Přidáním nových procesů by bylo možné celou analýzu ještě více paralelizovat.

6 Závěr

V rámci mé diplomové práce jsem se zabýval systémem zpracovávajícím a rozpoznávajícím trojrozměrné objekty pomocí deformačních gramatik. Hlavním cílem bylo navrhnout a otestovat koncept, jenž by si dokázal poradit s deformacemi objektu vzniklými neúplným popisem objektu a rozpoznat objekt v co možná nejkratším čase.

Prvním problémem, se kterým jsem se musel vypořádat, byla nutnost převést pole hodnot vzdáleností od středu do jiné podoby. Bez této úpravy nebylo možné jednoznačně přiřadit terminální symboly prvkům vstupního pole.

Rozhodl jsem se převést hodnoty vzdáleností na body v trojrozměrném souřadném systému. Spojením sousedních bodů pomyslnými přímkami lze získat polygonovou reprezentaci objektu. Jednotlivým plochám a hranám navíc lze přiřadit terminály. Další výhodou je možnost užití analytické geometrie, pomocí níž jsem mohl zjistit normálové vektory ploch, jež jsem použil na zjištění vzájemného úhlu mezi těmito plochami.

Na základě zmíněných úprav jsem navrhl maticovou gramatiku popisující trojrozměrné deformované objekty. Dále jsem představil sérii úprav, pomocí kterých jsem převedl tuto gramatiku na bezkontextovou. Ta by byla bezpochyby funkční, nicméně celou záležitost komplikovala její přílišná složitost.

Vypustil jsem tedy plochy z popisu objektu, což mělo za následek určité zjednodušení gramatiky a omezení počtu pravidel. Výsledkem byla funkční verze gramatiky, nicméně i tato byla značně složitá, navíc u otočených objektů nebylo možné snadno agregovat dílčí hrany do jedné souvislé.

Nakonec jsem se rozhodl pro popis objektu z osmi svislých a jedné vodorovné snímací roviny. Úhly mezi rovinami jsou počítány z pohledu popisné roviny, což má za následek pravdivější popis deformovaného objektu.

Mým cílem bylo co možná největší zjednodušení gramatiky. Omezením počtu produkčních pravidel dojde i k omezení možných nejednoznačností gramatiky. Výsledný syntaktický analyzátor pak dokáže pracovat rychleji a je daleko přímočařejší. Proto jsem navrhl samostatné zpracování jednotlivých rovin. Automat se zbaví zbytečných závislostí, dojde k výraznému omezení eventualit v LR tabulce, navíc jednotlivé stavy odpovídají určité fázi vstupu.

Celé snažení jsem ještě podpořil zavedením podmíněných pravidel, která dále přispívají k redukci nejednoznačností, neboť se automat nemusí zabývat chybnými odhady.

Pro zpracování navržené gramatiky jsem použil osvědčenou implementaci Tomita parsru, kterou jsem upravil pro potřeby nově navržené gramatiky. Předně bylo třeba změnit samotný styl řízení automatu, protože je nutné vykonat syntaktickou analýzu několikrát v rychlém sledu za sebou. Dále jsem přepracoval lexikální analyzátor a doplnil jsem metody nutné pro načtení a transformaci vstupního pole.

Nakonec jsem použil jednoduchý finální parser, který všechny dílčí výsledky spojí a definitivně rozhodne o klasifikaci objektu. Tento automat je založen na re-

kurzivním sestupu, jejž jsem mohl zvolit na základě dobrých výsledků dílčích syntaktických analýz.

Při závěrečném testování jsem odhalil, že mnou vytvořený systém pracuje velice rychle a dosahuje přesných výsledků. Rychlost analýzy vyplývá z úprav gramatiky a její celkové jednoduchosti. Redukce nejednoznačností přímo úměrně omezila i počet stavů, kterými automat během analýz musí projít.

Nevýhodou celého systému je určitá nerozšiřitelnost LR tabulky, respektive její přílišná specializace. Rozhodovací tabulka vychází z gramatických pravidel. Při drobných úpravách a rozšířeních je relativně jednoduché nové stavy doplnit. Rozsáhlejší změny si ale vyžádají její kompletní přepracování, což si vyžaduje značné úsilí.

Specializace gramatiky se pak projeví neschopností systému rozpoznat objekty, které nejsou popsány derivačními pravidly. Všechny známé objekty a jejich deformace jsou a budou korektně rozpoznány, u ostatních však dochází k selhání systému.

Případné zobecnění systému by si vyžádalo kompletní přepracování celého konceptu syntaktické analýzy a opuštění současných standardních postupů. Další možnou úpravou je paralelizace při rozpoznávání jednotlivých rovin, neboť proces probíhá zcela samostatně.

7 Literatura

1. ŠŤASTNÝ, JIŘÍ. *Netradiční metody a algoritmy pro rozpoznávání objektů technologické scény.* Brno : Vutium, 2006. ISBN 80-214-3117-2.
2. MEDUNA, ALEXANDR. *Automata and Languages: Theory and Applications.* London : Springer, 2010. 1-85233-074-0.
3. CHOMSKY, NOAM. *Three models for the description of language.* místo neznámé : IRE Transactions on Information Theory 2, 1956, Sv. 113-124.
4. GRUNE, DICK A CERIEL, J., H. JACOBS. *Parsing Techniques.* Chichester : Ellis Horwood, 1990. 978-0387202488.
5. MINAŘÍK, MARTIN. *Strukturální metody identifikace objektů pro řízení průmyslového robotu.* VUTBR. [Online] Červenec 2009. [Citace: 15. Únor 2014.] https://dspace.vutbr.cz/xmlui/bitstream/handle/11012/16254/Thesis%20Final.pdf?sequence=2.
6. *Hamming Distance.* School of Mathematics. [Online] The University of Manchester. [Citace: 20. Duben 2016.] http://www.maths.manchester.ac.uk/~pas/code/notes/part2.pdf.
7. *Euclidian Distance.* The Math Explorers' Club. [Online] Cornell University. [Citace: 20. Duben 2016.] http://www.math.cornell.edu/~mec/Summer2008/Jones/euclid.htm.
8. FELZENSZWALB, F., PEDRO A MCALLESTER, DAVID. *Object Detection Grammars*. Brown University. [Online] 11. Únor 2010. [Citace: 25. Březen 2014.] http://cs.brown.edu/~pff/papers/TR-2010-02.pdf.
9. MCLEAN, PHILIPPE A HORSPOOL, NIGEL. *A Fast Early Parser.* [Online] [Citace: 20. Duben 2014.] http://webhome.cs.uvic.ca/~nigelh/Publications/fastEarley.pdf.
10. ANDERSON, L., DAVID A SHAPIRO, LIONEL. *Introduction to Chain Code.* [Online] 2006. [Citace: 20. Duben 2014.] http://www.mind.ilstu.edu/curriculum/chain_codes_intro/chain_codes_intro.php.
11. SIDDIQI, KALEEM A KIMIA, B., BENJAMIN. *A Shock Grammar For Recognition.* Researchgate. [Online] IEEE Xplore, 1996. [Citace: 20. Duben 2016.] https://www.researchgate.net/profile/Kaleem_Siddiqi/publication/3637793_Shock_grammar_for_recognition/links/00b4952537acb6591e000000.pdf.
12. ZHU, SONG-CHUN A MUMFORD, DAVID. *A Stochastic Grammar of Images.* Dgital Acces To Scholarship At Hardvard. [Online] Foundations and Trends in Computer Graphics and Vision 2, 2006. [Citace: 20. Duben 2016.] https://dash.harvard.edu/handle/1/3637153.
13. SUJATHAKUMARI, K. A DERSANAMBIKA, K., S. *Cooperating distributed context-free hexagonal array grammar systems with permitting context.* International Journal of Mathematics Trends and Technology. [Online] International Journal of

Mathematics Trends and Technology, Březen 2014. [Citace: 20. Duben 2016.] http://arxiv.org/abs/1404.5393. ISSN: 2231-5373.

14. BELONGIE, SERGE, MALIK, JITENDRA A PUZICHA, JAN. *Shape Matching and Object Recognition Using Shape Context.* IEEE Transactions on Pattern Analysis and Machine Intelligence. [Online] IEEE, Duben 2002. [Citace: 20. Duben 2016.] https://staff.fnwi.uva.nl/r.vandenboomgaard/UvAwiki/uploads/HVisser/Shape_Matching_and_Object_Recognition_Using_Shape_Contexts_Belongie_Malik_Puzicha.pdf.
15. LOWE, G., DAVID. *Three-dimensional object recognition from single two-dimensional images.* Artificial Intelligence. [Online] Elsevier Science Publishers Ltd. Essex, UK , Březen 1987. [Citace: 20. Duben 2016.] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.145.5388&rep=rep1&type=pdf.
16. JUNEK, LUKÁŠ. *Návrh systému pro aplikaci deformačních gramatik.* [Online] 23. Květen 2014. https://is.mendelu.cz/auth/zp/zp_portal.pl?studium=55850;obdobi=356.

Přílohy

A Elektronické dokumenty na CD

K práci je přiloženo CD, které obsahuje následující soubory:

- Zdrojové soubory implementovaného systému
- Elektronická verze tohoto dokumentu
- Makefile pro přeložení implementovaného systému
- Soubor s nápovědou k programu
- Soubory se vstupním polem hodnot vzdáleností pro jednotlivé objekty

B Seznam vstupních souborů

Přehled testovaných objektů a vstupních souborů (všechny mají příponu txt):

- Koule: koule, koule-mala, koule-velka
- Kvádr: kvadr-a, kvadr -b, kvadr -c, kvadr -d, kvadr -e, kvadr -f
- Osmistěn: osmisten, osmisten-45x, osmisten-45y, osmisten-90z-a, osmisten-90z-b, osmisten-90z-c
- Obrácený osmistěn: obrac-osmisten, obrac-osmisten-45y, obrac-osmisten-90z
- Válec s prohnutým pláštěm: proh-valec-1, proj-valec-2, proh-valec-3, proh-valec-4
- Vázy: vaza-tvar1, vaza-tvar2, vaza-tvar2-mala

Testované vlastnosti: u většiny objektů jsem testoval, jak se otočení objektu nebo změna jeho velikosti projeví na deformaci tohoto objektu. Dochází ke změně pozice klíčových úhlů, rozbití úhlu na více menších, popřípadě ke změně velikosti klíčového úhlu.

Především u různých velikostí koulí a válců jsem zkoumal, jak se změní velikosti dílčích úhlů ve snímané rovině.

Naopak u různě natočených kvádrů jsem testoval schopnost lexikálního analyzátoru správně vyrovnat normálové vektory do roviny snímání.

U obrácených osmistěnů a prohnutých válců jsem posuzoval schopnost analyzátoru rozlišit konkávní tvar prohnutí pláště.