VĚDECKÉ SPISY VYSOKÉHO UČENÍ TECHNICKÉHO V BRNĚ

Edice PhD Thesis, sv. 727

ISSN 1213-4198

Ing. Jiří Lýsek

Rozpoznávání objektů pomocí evolučních metod

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

Ing. Jiří Lýsek

ROZPOZNÁVÁNÍ OBJEKTŮ POMOCÍ EVOLUČNÍCH METOD

OBJECT RECOGNITION BY MEANS OF EVOLUTIONARY METHODS

Zkrácená verze Ph.D. Thesis

Obor:	Konstrukční a procesní inženýrství
Školitel:	prof. RNDr. Ing. Jiří Šťastný, CSc.
Oponenti:	doc. Ing. Ivana Rábová, Ph.D. doc. Ing. Vladislav Škorpil, CSc.
Datum obhajoby:	20. listopadu 2013

Klíčová slova
Evoluční metody, počítačové vidění, rozpoznávání, učení.

Keywords
Evolutionary methods, computer vision, recognition, learning.

Místo uložení práce:
Knihovna VUT FSI v Brně

ISBN 978-80-214-4875-9
ISSN 1213-4198

OBSAH

1 ÚVOD

Evoluční metody jsou počítačové algoritmy inspirované procesem evoluce, který probíhá všude kolem nás již miliardy let. Tyto metody nehledají inspiraci v nějakém hotovém řešení, které se v přírodě nachází, ale napodobují samotnou podstatu evolučního procesu. Tedy přírodou navrženého hledání optimálních řešení.

Jde tedy o přechod genetických informací uložených v rodičích na jejich potomky pomocí procesu křížení. Dále o náhodnou mutaci, která modifikuje předané genetické informace a vliv prostředí a schopnost nového jedince v daném prostředí existovat a přežít. Tato schopnost každého organismu musí být nějakým způsobem ohodnocena - v přírodě je to délka a kvalita života takového jedince, v počítačové simulaci můžeme vhodnost jedince ohodnotit pouhým číslem.

V dnešní době roste potřeba algoritmů, které by uměly řešit mnoho typů úloh a nebylo by potřeba je příliš složitě upravovat a ladit pro nové aplikace. Takové algoritmy by mohly být nasazeny k řešení nejrůznějších typů problémů, kde zatím neznáme optimální přístupy, nebo nemáme dostatek prostředků k použití lidských expertů. Evoluční metody mohou takový rámcový systém pro řešení problémů v mnoha doménách poskytnout.

Univerzálnější přístup nabízí gramatická evoluce, která dokáže využít standardní princip optimalizace založené na evolučním principu a spojit jej s doménou řešeného problému prostřednictvím definované gramatiky a pomocí hodnotící funkce, která slouží k výpočtu vhodnosti jedince pro řešení daného problému. Gramatická evoluce je dnes úspěšně nasazována na problémy generování různých matematických modelů, kde může dosahovat lepších výsledků, než klasické metody.

U evolučních metod je ale nutné si uvědomit, že jde o úplně jiný přístup k řešení problému. Výsledek evoluční metody vždy silně závisí na parametrech a možnostech algoritmu. Někdy evoluční metoda není schopna přinést lepší řešení problému po mnoho iterací výpočtu. Potom najednou přijde díky náhodě nové řešení a tím se celý proces zlepšování řešení problému zase rozhýbe. Evoluční metody pracují velmi často s nepředstavitelně velkým prostorem řešení, který má mnoho lokálních extrémů. Díky tomuto je někdy nutné celý proces spouštět opakovaně, aby bylo nalezeno to nejlepší řešení.

Dalším intenzivně zkoumaným tématem je využití metod umělé inteligence k zastoupení lidí. Konkrétně snaha naučit počítače a roboty vnímat – rozpoznávat. Důvody k nasazení umělé inteligence jsou různé, ale výsledkem by mělo být vždy zlepšení proti předchozímu stavu. Proto je nutné vylepšovat a hledat metody, jak stroje naučit efektivně pomáhat.

V této práci bude gramatická evoluce použita jako nástroj pro hledání spustitelných programových struktur, které na základě vstupních dat budou schopné identifikovat předložené objekty.

1.1 CÍL PRÁCE

Cílem práce je navrhnout variantu genetických algoritmů pro aplikaci v oblasti rozpoznávání objektů. Jde tedy o aplikaci evolučních metod v oblasti, kde tradičně používáme buď klasické statistické metody, nebo zaběhnuté metody umělé inteligence, jako jsou například neuronové sítě. Práce je zaměřena hlavně na aplikaci gramatické evoluce, která je vhodná k vytváření struktur, které lze spouštět jako počítačové programy. Abychom takovéto struktury získali a byly užitečné, je nutné pomocí evolučního procesu tyto programy nejprve nechat vyvinout. Tento proces se dá

přirovnat k učení neuronové sítě před jejím nasazením k řešení úkolu. I pomocí evoluční metody vlastně do programové struktury ukládáme informace, které pak slouží k vyřešení daného problému. Na rozdíl od neuronové sítě však tato struktura není nikdy pevně daná a její parametry jsou dopředu neznámé.

Ke splnění tohoto cíle je nutné vytvořit testovací prostředí, kde bude možné spouštět různé varianty evolučních algoritmů. V tomto prostředí potom budou testovány různé možnosti jak využít evoluční metody k rozpoznávání objektů v obraze.

2 ROZPOZNÁVÁNÍ OBJEKTŮ V OBRAZE

Rozpoznávání objektů je široký obor s mnoha úspěšnými aplikacemi. V dnešní době hojného rozšíření levných digitálních kamer, integrovaných do všemožných zařízení, má nějakou aplikaci počítačového vidění k dispozici každý majitel takovéhoto zařízení (například detekce obličejů ve fotoaparátu, rozpoznávání a autorizace uživatele PC pomocí kamery, …). Tyto aplikace sloužící převážně k zábavě jsou spíše vedlejším produktem aplikací s důležitou rolí v průmyslu, bezpečnosti, vojenství, zdravotnictví a jiných oborech, na kterých můžou záviset ekonomiky států nebo bezpečnost obyvatel. Pro aplikace metod počítačového vidění jsou tudíž hlavní kritéria rychlost a spolehlivost.

S rozvíjejícím se výpočetním výkonem počítačů je možné aplikovat stále sofistikovanější metody pro rozpoznávání objektů. Někdy však požadavek na rychlost rozpoznání stojí proti požadavku na spolehlivost a je nutné najít vhodný kompromis.

2.1 DŮLEŽITÉ PARAMETRY PROCESU

Proces rozpoznávání má podle své aplikace a použité metody následující důležité vlastnosti. Podle aplikace samozřejmě vyžadujeme různé kombinace nebo míru těchto parametrů. Pokud se uvažuje například průmyslová aplikace pro rozpoznávání víceméně plošných objektů, bude důležitým parametrem rychlost a spolehlivost. Podle typu objektů budeme vyžadovat ještě invarianci vůči rotaci, či stranovému převrácení.

Často je při nasazování metod umělé inteligence zároveň nutné upravit i výrobní proces. Například zavedením určitých značek na objektech ve výrobních linkách.

2.1.1 Spolehlivost

Hlavním úkolem algoritmů pro rozpoznávání objektů je spolehlivost. Proto se u všech metod nejčastěji sleduje právě tento parametr.

2.1.2 Rychlost

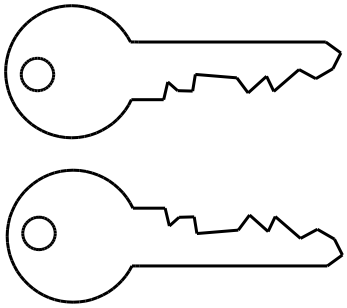
V některých aplikacích je důležité, aby proces rozpoznání proběhl co nejrychleji. Rychlost ale může být vykoupena nesplněním některých dalších parametrů procesu. Rychlost procesu je také často faktorem, který ovlivní cenu celého řešení. Ať už na nakoupeném hardware nebo na nákladech na vývoj a optimalizaci metody.

2.1.3 Invariance vůči rotaci

Často kladeným požadavkem je, aby metoda nebyla citlivá na orientaci objektů, které mají být rozpoznány. Je to logický požadavek, neboť například na dopravníkovém pásu nikdo nezaručí, že objekty budou přicházet vždy srovnané.

2.1.4 Invariance vůči stranovému převrácení

Tato vlastnost je dobře demonstrována na obrázku klíče 2.1. Pokud naše metoda umožní rozlišení horního a dolního klíče, je invariantní vůči stranovému převrácení.



Obrázek 2.1.: Ukázka stranově převrácených objektů

2.1.5 Invariance vůči změně měřítka

Tato vlastnost může být někdy nežádoucí. Jde o to, že by metoda měla být schopna rozpoznat stejné objekty i v různých vzdálenostech od snímače.

2.2 REPREZENTACE OBRAZU

Obrazová informace je v počítači uchována jako číselná matice. Každá hodnota v této matici odpovídá jednomu bodu, který může být zobrazen na obrazovce počítače – tento počet bodů nemusí být nutně stejný, jako počet bodů snímače použitého pro pořízení obrazu. První důležitou informací je tedy velikost této matice.

Dalším kritériem je počet těchto matic. Pro uložení obrazu se používají obvykle takovéto systémy:

- 1 matice – stupně šedi, binární obraz
- 3 matice – systémy *RGB, HSV*, …
- 4 matice – systémy *RGBA, CMYK*, …

Každá matice reprezentuje jednu složku obrazu.

Nejčastěji se používá pro ukládání obrazu *RGB* barevný model s 8 bity na barvu, který je aditivní a je složený ze tří složek R – červená (Red), G – zelená (Green) a B – modrá (Blue). Míchání třech základních složek barvy získáme 256^3 barev.

Pokud volíme reprezentaci obrazu ve stupních šedi, pak se obvykle používá rozdělení na 256 stupňů – tedy 8 bitů na každou hodnotu matice. V počítačovém zpracování obrazu velmi často pracujeme právě s obrazem ve stupních šedi, jelikož nám dovoluje pracovat s menším objemem dat, než při práci s barevným obrazem.

Pokud chceme převést barevný obraz na obraz ve stupních šedi, existuje snadný přepočet. Někdy je tento přepočet nutný i z důvodů, že výstupní zařízení (například tiskárna) nepodporuje barevný výstup.

2.3 PŘEDZPRACOVÁNÍ

Většina aplikací pro rozpoznávání objektů v obraze používá nějaké předzpracování obrazu pro zlepšení výsledků dané metody. Všechny kroky předzpracování je nutné provádět s rozvahou, abychom použité metodě pro popis objektů a rozpoznání nezabránili správně fungovat.

2.4 DETEKCE MÍST ZÁJMU, SEGMENTACE

Následujícím krokem může být vyhledání oblastí obrazu, na které následovně budeme rozpoznávat objekty. Tento krok často klade důraz na získání částí obrazu – segmenty, které potenciálně obsahují nějaké objekty a odstranění oblastí obrazu, kde je pozadí. V podstatě se jedná o binární klasifikaci.

Segmentaci lze přeskočit, pokud použijeme posuvné okno, které si prohlédne celý obraz krok po kroku.

2.4.1 Prahování

Pomocí prahování můžeme oddělit světlé a tmavé oblasti obrazu. Obvykle se pracuje s předem nastavenou hodnotou pro oříznutí. Sofistikovanější aplikace pracují s histogramem obrazu. Pomocí prahování lze obraz segmentovat.

Prahování pomocí histogramu lze provádět i dynamicky na menších výřezech z obrazu.

2.4.2 Hrany a rohy

Existují postupy, pomocí kterých lze z obrazu získat místa, kde jsou hrany (prudké změny gradientu). Jsou to například Sobelův operátor, Canny edge detector a další. Pracují obvykle na principu konvoluce obrazu a specifického konvolučního jádra. Další obvyklé detektory jsou určeny k detekci rohů (křížící se hrany). Pomocí nalezených hran lze obraz opět segmentovat. Získání popisu objektu

Pokud vlastní rozpoznání objektů nepracuje přímo na obrazové matici, je nutné z obrazových segmentů získat popis objektů. Například získat statistické charakteristiky o barvě nebo pomocí frekvenční analýzy informace o frekvencích v obraze. Další možností je využít hrany objektů pro získání popisu kontury objektu. Popis objektů je nutné navrhnout specificky podle řešené úlohy.

2.5 VLASTNÍ ROZPOZNÁNÍ – PŘEHLED METOD

Vlastní rozpoznávání objektů můžeme realizovat pomocí metod umělé inteligence nebo statistickou analýzou. Obvykle dojde k předložení získaného popisu objektu tzv. klasifikátoru, který s určitou mírou přesnosti dokáže určit, o který objekt se jedná. Klasifikátor je dopředu seznámen s objekty, které mu mohou být předkládány. Tento proces se nazývá učení. Pokud je mu předložen neznámý objekt, obvykle nedokáže upravit naučené znalosti, aby jej dokázal rozpoznat. Existují samozřejmě systémy, které se dokážou adaptovat.

2.5.1 Běžné metody umělé inteligence

Pro rozpoznávání libovolných objektů se nejčastěji používá neuronová síť. Je to struktura tvořená umělými neurony [24], které představují model lidského neuronu a tyto jsou navzájem různě propojeny. Podle systému propojení se neuronové sítě dělí na dopředné a se zpětnou vazbou. Rozložení neuronů a jejich propojení se souhrnně nazývá topologie.

Pro rozpoznávání se často používají dopředné vícevrstvé sítě (*MLP*). Neurony jsou v nich uspořádány po vrstvách a každá vrstva má přístup k výstupním hodnotám předchozí vrstvy. Pro učení takovýchto sítí se používá algoritmus backpropagation [36].

Dalším klasifikačním nástrojem může být Kohonenova samoorganizační mapa [17]. Tato síť pracuje na principu shlukování podobných vzorů. Dá se tedy použít k automatické klasifikaci bez učitele.

Dnes již zastaralou, sítí je Hopfieldova síť [16], která dokáže rozpoznávat pouze binární vstupní vzory.

Dalším možným nástrojem pro klasifikaci jsou tzv. *Support Vector Machines* (*SVM*) [9]. A mnoho jiných.

V neposlední řadě je možné vytvářet různé rozhodovací stromy nebo jiné modely založené i na obecné popisné statistice.

2.5.2 Přístupy založené na evolučních metodách

V současnosti již byly evoluční algoritmy více či méně úspěšně použity k řešení různých úkolů k týkajících se rozpoznávání objektů v obraze. Některé aplikace používaly evoluční metody jen jako podporu k řešení vlastního problému například pomocí neuronových sítí. Například v [6] jde o optimalizaci dat použitých k učení klasifikátoru.

Jiné aplikace využívají genetické algoritmy přímo k operacím vedoucím ke klasifikaci objektů. Tyto lze rozdělit podle použité varianty evoluční metody na implementace se specifickým chromozomem a ty, které používají gramatickou evoluci.

Metody využívající specifickou interpretaci chromozomu

Tyto metody obvykle vyžadují více informací o řešené úloze. Programátor musí vymyslet, jak bude uložen chromozom, jak bude probíhat křížení a mutace a jak budou jedinci interpretování a hodnoceni. Výsledkem je tedy systém, který není použitelný univerzálně, ale pouze k řešení jednoho typu úlohy. Výhodou může být větší rychlost navrženého systému pro řešení požadované úlohy.

U těchto přístupů jde většinou o přiřazování vybraných charakteristik z předloženého obrazu k charakteristikám uloženým v databázi modelů.

Přiřazování kontury objektu k modelu

Dobré výsledky jsou dosaženy pro přiřazování kontury objektu z databanky ke kontuře získané například detekcí hran. Chromozom jedince může kódovat třídu objektu z databanky, jeho natočení, pozici jeho těžiště apod. Další možná informace může být zkosení, zúžení čí zploštění objektu.

Pro vyhodnocení fitness se obvykle používá výpočet vzdáleností bodů scény modelované genetickým algoritmem od bodů získaných hranovým detektorem. Čím je větší shoda modelu s předloženou scénou, tím je vzdálenost všech bodů menší.

Pokud ve scéně předpokládáme více než jeden objekt, je možné genetický algoritmus spustit vícekrát a postupně odmazávat nalezené objekty.

Výhoda tohoto přístupu je hlavně ve schopnosti rychlé adaptace genetických algoritmů na proměnlivé prostředí, takže není problém aplikovat takovouto metodu i pro pohyblivou scénu.

Někdy jsou pomocí genetického algoritmu přiřazována jen geometricky primitivní tvary (kruhy, čtverce apod.) jako v [34]. Detekce jednoduchých tvarů, lze použít například pro detekci dopravních značek, jako v [14]. Někdy je v databance objektů uložena informace o kompletní kontuře objektu dokonce i ve více možných pohledech. Takový postup byl použit např. v [4], [43], [12].

Přiřazování jiných charakteristik k modelu

Podobně jako přiřazování kontury lze pomocí evolučních metod přiřazovat jiné charakteristiky získané z obrazu k charakteristikám uloženým v databázi modelů. Např . v [40] je genetický algoritmus použit k přiřazování uzlů grafu obsahujícího atributy obrazu.

Metody využívající gramatickou evoluci

Výhodou použití gramatické evoluce v jakékoliv aplikaci evolučních metod je oddělení procesu evoluce od interpretace jedinců pro konkrétní řešenou úlohu. Gramatická evoluce vidí jedince jako programy generované gramatikou (což je samozřejmě interpretace číselného chromozomu). Tato interpretace dovoluje provádět standardní operace mutace a křížení, což algoritmu stačí k tvorbě nových jedinců a nepotřebuje k tomu znalosti o řešeném problému.

Gramatickou evoluci nezajímá, k čemu je program nakonec použit. Pro problém řešený pomocí gramatické evoluce tedy potřebujeme nadefinovat pouze gramatiku a hodnotící funkci pro získání hodnoty fitness každého jedince.

Do jisté míry se tedy dá říct, že gramatická evoluce je obecný nástroj pro tvorbu jakýchkoliv počítačových programů či jiných struktur, které jsou generovány gramatikou.

Použití těchto metod je lákavé hlavně proto, že slibuje aplikaci v problémech, kde dopředu neznáme informace o předkládaných objektech. Vhodné by mohlo být například využití k automatické klasifikaci velkého množství objektů na obrazech (například v prostředí internetu), kde by evoluční metoda mohla nalézt společné charakteristiky velkého množství označených objektů na poměrně malém vzorku obrazů. Následně by mohlo probíhat vyhledávání obrazů s podobnými charakteristikami v určitých lokacích a tyto výsledky by se použily jako odpověď na vyhledávací dotaz.

Metody založené na posuvném okně

Nejobecnější přístup k rozpoznávání objektů v obraze mají metody založené na přístupu posuvného okna. U těchto metod nedochází v podstatě k žádné extrakci dat z obrazu. Matice obrazových bodů je přímo předložena nástroji pro rozpoznání třídy objektu.

U přístupu využívající posuvné okno je nutné postupně předložit klasifikátoru celý obraz a to může v závislosti na složitosti klasifikátoru trvat poměrně dlouho. Jediný parametr, kterým lze u této metody ovlivnit rychlost procházení obrazu je krok okna. Pokud se posuvné okno nepohybuje bod po bodu, ale vynechává každý druhý bod, sníží se doba nutná k průchodu celého obrazu 4×. Čím větší krok, tím méně času je nutné pro zpracování celého vstupu.

Někdy jsou tyto metody používány jen k detekci objektů (klasifikace do 2 tříd). Tento přístup byl využit například k detekci obličejů v [44].

Jindy jsou evoluční metody použity k tvorbě programů, které dokáží klasifikovat objekty do více tříd [46][45]. Zde je potom velký problém s nastavením výstupních hodnot jedinců (programů), které obvykle vrací jen desetinné číslo. Naproti tomu neuronové sítě dokážou na svém výstupu vrátit celý vektor, který se snadněji interpretuje.

Použití gramatické evoluce k získání lokálních deskriptorů

Gramatickou evoluci je možné použít k získání lokálních deskriptorů podobných těm, které dostáváme při použití metody SIFT [20] nebo SURF [2]. V [28] je popsán postup získání krátkých programů, které označují místa k vypočtení lokálních deskriptorů. Tyto deskriptory, ať už získané

pomocí kterékoliv metody, mohou být použity k přiřazování odpovídajících míst na předloženém obraze k místům uloženým v nějaké databázi a tím můžeme dosáhnout rozpoznání objektů.

Generování L-systémů pomocí gramatik

L-systém [33] (Lindenmayerův systém) je paralelní přepisovací gramatika, která je schopná modelovat fraktály. Pokud tuto gramatiku poskytneme gramatické evoluci, budeme dostávat různé řetězce. Tyto jsou obvykle znázorněny graficky pomocí tzv. želví grafiky a porovnány s předloženým obrazem.

Pro výpočet fitness hodnoty se použije podobný postup jako při postupu s přikládáním kontur.

Příklad použití tohoto postupu je v [3] a [27], kde jsou pomocí L-systémů hledány vhodné aproximace předložených tvarů.

Někdy je tento postup použit k vytváření obrazů v počítačové grafice. Není zde pak automatická hodnotící funkce, ale člověk, který řadí vygenerované obrazce podle toho, jak se mu jednotliví jedincí vizuálně líbí.

2.5.3 Další příklady použití evolučních metod v oblasti počítačového vidění

Evoluční metody jsou často používány k řešení podpůrných úloh, které přímo nesouvisí s klasifikací objektů. Velmi dobré výsledky jsou dosahovány u problémů segmentace obrazu. Jedinci obvykle kódují rozložení regionů nad obrazem (určují příslušnost obrazových bodů do regionů).

Optimalizovaná hodnota pak může být například co nejmenší rozdíl v nějaké sledované charakteristice (barva, textura, …) v oblastech spadajících do jednoho regionu proti co největší rozdílnosti v odlišných regionech. Další optimalizované kritérium může být délka hranic regionů. Příklady se dají nalézt v [13][10][39][5][38].

Další zajímavá aplikace je k nalezení v [19], kde je genetický algoritmus použit k získání informací o rozdílech ve stereo obrazech. Toto se dá využít například k sestrojení 3D mapy terénu z leteckých fotografií.

3 NAVRŽENÉ ALGORITMY

Hlavním cílem této práce navrhnout aplikaci evolučních metod [18], konkrétně gramatické evoluce [37], v oblasti počítačového vidění – rozpoznávání objektů. Jinými slovy to znamená použití gramatické evoluce k tvorbě klasifikátoru. Jde tedy o proces učení a výsledkem by měl být systém, který by mohl pracovat podobně jako neuronová síť.

Samotné evoluční metody nejsou přímo vhodné jako ekvivalent neuronové sítě. Běh evoluční metody je iterativní optimalizační proces, podobající se spíše procesu učení neuronové sítě.

Neznamená to, že by evoluční metoda nebyla použitelná k rozpoznávání objektů. Pomocí evoluční metody lze například generovat polohu objektu a pomocí fitness kontrolovat míru překrytí kontury z databáze s hranou detekovanou hranovým detektorem. Evoluční algoritmus ovšem nezaručuje vždy stejný průběh a konvergence ke globálnímu extrému také není vždy zaručena. Problémem evolučních metod tedy je, že nezaručují opakovaně stejné hodnoty na výstupu při stejných hodnotách na vstupu.

Další nevýhodou je nutnost ohodnocování velkého množství jedinců funkcí pro výpočet fitness hodnoty. Tato akce může hlavně u aplikací počítačového vidění trvat dlouho. Postavení evoluční metody jako alternativy k neuronové síti v produkční fázi se tedy nejeví jako dobrý nápad.

Je proto vhodnější genetický algoritmus použít ke tvorbě klasifikačního nástroje. Zde se nabízí použití právě gramatické evoluce, která je schopná vytvářet stromové programy. Struktura, terminály gramatiky a konstanty použité k sestavení stromu se dají považovat za znalosti. Při použití vhodné funkce pro výpočet fitness hodnoty je pak možné řídit optimalizační proces k tvorbě výkonných klasifikátorů.

Předpokládaná aplikace metod je v průmyslu, například na automatické výrobní lince, kde hlavními požadavky jsou spolehlivost a rychlost rozpoznání.

3.1 POŽADAVKY

Z předchozího textu vyplývá, že hlavním úkolem bude nalezení způsobu, jak pomocí evolučních metod vytvářet klasifikátory schopné rozpoznávat předložené objekty.

Hlavním požadavkem je univerzálnost použití navržené metody. Univerzálností se myslí, že metoda bude použitelná nejen pro rozpoznávání objektů v obraze, ale k rozpoznávání jakýchkoliv objektů, které lze popsat číselně a tyto data zpracovat v počítači. I toto má samozřejmě svá omezení – schopnost úspěšného řešení problému pomocí gramatické evoluce vždy závisí na prostředcích, které dáme optimalizačnímu procesu k dispozici. Je to jednak počet iterací a jedinců v populaci, ale hlavně sada zvolených terminálů.

Dalším vytyčeným požadavkem bude rychlost, která by měla být srovnatelná s neuronovou sítí.

V neposlední řadě pak spolehlivost. Je samozřejmé, že v každé aplikaci počítačového vidění záleží na nastavených podmínkách, nicméně není přípustné, aby měl klasifikátor jinou odezvu pro stejná vstupní data. Proto byl zavržen přístup s aktivní fází založenou na evoluční metodě.

3.2 ZVOLENÁ ARCHITEKTURA

Jak již bylo řečeno, v práci je použita gramatická evoluce. Klasifikátorem tedy bude program vytvořený tímto postupem. Evoluční metoda bude použita v učící fázi. Dalším krokem je zvolit, jak budou reprezentovány učící a testovací vzory.

Bylo zvoleno, že se bude pracovat s obrazem v odstínech šedi.

3.2.1 Princip posuvného okna

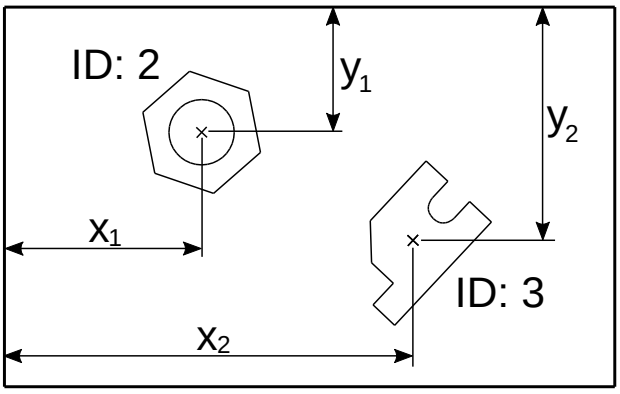
Universálním přístupem je přímé předkládání částí obrazu klasifikátoru. Tzv. posuvné okno je jakýsi výřez z velké obrazové matice. Hodnoty pixelů posuvného okna lze snadno dosadit jako vstupní hodnoty do klasifikátoru.

3.2.2 Vektorový nebo maticový vstup

Jako vstup bude tedy napojeno posuvné okno, pro univerzálnost je toto definováno jako matice. Pro jiné využití lze matici degradovat na vektor a použít úplně stejný postup pro tvorbu klasifikátoru.

3.2.3 Příklady pro učení a testování

Pro učení a testování metod je nutné nachystat i odpovídající data. Protože má jít o rozpoznávání objektů v obraze, je vstup obraz typu PNG nebo JPG s nepřekrývajícími se objekty v různých polohách. K těmto obrázkům je dále připojena informace o pozici objektů a jejich třídě. Vše je uloženo v textovém souboru. V obraze se teoreticky nemusí vyskytovat žádný objekt, je tam tedy pouze pozadí. V takovém případě v popisném souboru nic není – pozadí se nepopisuje.



Obrázek 3.1.: Ukázka dat pro učení a testování

3.2.4 Vyhodnocení výsledků klasifikace

Vzhledem ke zvolenému přístupu s posuvným oknem, je pro každou pozici, na které se posuvné okno zastavilo, k dispozici jeden výsledek klasifikace. Výsledek klasifikace lze tedy získat analýzou shluků bodů. Nejprve se odfiltrují výsledky, kde je podle klasifikátoru pozadí. Poté by měly zůstat pouze samostatné skupiny bodů. Pro každou skupinu pak snadno určíme třídu objektu jako nejčastěji vyskytující se hodnotu (modus).

Případně lze nalézt těžiště shluku a jako výsledek klasifikace získat nejčastější hodnoty v jeho okolí. Tato metoda je vhodnější, pokud okolo objektů vzniká mnoho neurčitých výsledků.

3.3 TERMINÁLY GRAMATIKY

Konkrétní gramatika [15], která by měla být předána procesu gramatické evoluce je závislá na doméně klasifikační úlohy. Pokud jde o rozpoznávání objektů v obraze s použitím posuvného okna, jak je popsáno v 3.2.1 a 3.2.2, bude určitě zapotřebí možnost pracovat s jednotlivými body obrazu. Proto je jasné, že v gramatice se musí objevit terminály, které dokážou číst oblasti v posuvném okně.

Dalšími vstupními terminály, pokud jde o obraz můžou být různé transformace, jako například prahovaný obraz nebo výstup některého hranového detektoru.

Abychom umožnili z těchto obrazových matic získat skalární hodnoty, budou nutné terminály pro výpočet základních operací jako suma, nebo průměr. Tyto mohou být kombinovány s terminály, které omezují regiony, nad kterými se odpovídající statistická operace vykoná.

Dále lze do gramatiky vložit terminály reprezentující základní matematické operace a funkce. Konkrétní gramatiky jsou v kapitole 5 , kde se nachází testy metod.

3.4 STROMOVÝ PROGRAM SE SKALÁRNÍM VÝSTUPEM

První variantou, která byla navržena, bylo použití stromového programu, který bude mít jako výstup jednu hodnotu. Inspirací byla práce [45]. Princip je takový, že výstup programu by měl mít určitou hodnotu, při předložení objektu určité třídy na vstup. Podle intervalu, do kterého hodnota patří se potom určila třída objektu na vstupu.

3.4.1 Přímá klasifikace

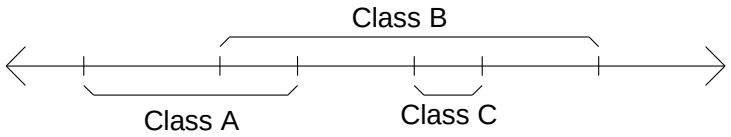
Jako první pokus s touto metodou byl tedy gramatickou evolucí vytvořen program, který přímo na obraze dokázal detekovat objekty. V následujícím odstavci je popsán princip získávání intervalů hodnot pro klasifikaci.

Statistické vyhodnocení výstupu programu

V citované práci [45] jsou intervaly pevně dané, což je poměrně tvrdá podmínka, která může proces gramatické evoluce zpomalit. Inovací v tomto případě byla automatická detekce intervalů.

Každý jedinec byl použit na trénovací množině obrazů. Podle výstupu programu v okolí známých objektů byl vypočten aritmetický průměr a směrodatná odchylka. Z těchto hodnot byly sestaveny hranice intervalů výstupu. Fitness funkce nehodnotila překrytí intervalů, ale pouze úspěšnost detekce. Jedinec, který měl vyhodnocené hranice intervalů tak, že se překrývaly, byl podle fitness funkcí hodnocen jako nekvalitní, jelikož prováděl chybné klasifikace.

Na obrázku 3.2 je naznačen problém s překrývajícími se intervaly.



Obrázek 3.2.: Program se skalárním výstupem - překrývání tříd

3.4.2 Dvoufázová klasifikace

Výsledky první metody nebyly příliš dobré, a proto bylo nutné přístup upravit. Výstup programu byl často správný v okolí středu objektu, ale program chyboval v blízkém okolí a na hranicích objektu. Proto bylo rozhodnuto o použití dvoufázového přístupu, kde prvním krokem bude detekce objektů v obraze a druhým krokem bude klasifikace pouze v místech označených detektorem.

Fáze 1: binární klasifikace – detekce objektů

Pro detekci byl použit opět program vytvořený gramatickou evolucí. Byl použit i přístup s automatickou volbou intervalů (jeden interval pro pozadí, druhý pro libovolný objekt).

Pro testovací objekty, které budou představeny v následující kapitole, by šlo použít místo detektoru i jednoduché prahování. Pro složitější objekty ve složitější scéně (například detekce obličejů na různých pozadích) je tento postup opodstatněný.

Detektor označil v obraze místa, kde byl zjištěn objekt. Klasifikátor byl spuštěn pouze v těchto bodech. Toto přineslo i poměrně velké zrychlení procesu klasifikace, neboť klasifikátor byl poměrně složitý program, jehož frekventované spouštění zabíralo mnoho času. Naproti tomu detektor byl velmi jednoduchý a rychlý.

Další urychlení procesu by bylo možné dosáhnout, pokud bychom detektor spouštěli s proměnlivým krokem okna. Tedy v oblasti, kde detektor reaguje pozitivně, bychom mohli krok posuvného okna zmenšit a v oblastech, kde nic není, zase zvětšit.

Fáze 2: klasifikace do více tříd

Druhá fáze již sloužila pouze k získání klasifikace objektů. Jak bylo uvedeno, klasifikátor se spouštěl pouze v označených místech. Zbytek metody byl stejný jako 3.4.1.

3.4.3 Výpočet funkce fitness

Pro klasifikátory byl vytvořen vzorec (3.1). Funkce pro výpočet hodnoty fitness zvyšuje svoji hodnotu s klesajícím počtem falešných positiv nebo negativ. Zároveň hodnota fitness funkce roste, pokud roste počet správně detekovaných bodů. Výsledek výpočtu leží v intervalu <0, 1>.

$$Fit(M) = \frac{C}{1 + C + FP + FN} \tag{3.1}$$

- C = počet správně detekovaných míst
- FP = počet falešných positiv (místa nesprávně označená jako objekt)
- FN = počet falešných negativ (místa, která měla být označená, ale nebyla)

Pokud byl použit dvoufázový přístup, výpočet fitness hodnoty pro detektor byl realizován vzorcem (3.2). V tomto vzorci je navíc část, která přidává požadavek na jednoduchost vytvořeného programu. Jinak je vzorec podobný vzorci (3.1), jen se nezajímá o počet správně detekovaných bodů. Výsledek výpočtu leží v intervalu <0, 1>.

$$Fit(M) = w_1 \frac{1}{1 + FP + FN} + w_2 \frac{1}{1 + NC} \tag{3.2}$$

- w_1, w_2 = váhy, nastaveny experimentálně
- FP = počet falešných positiv (místa nesprávně označená jako objekt)
- FN = počet falešných negativ (místa, která měla být označená, ale nebyla)
- NC = počet uzlů grafu (vyjadřuje složitost programu)

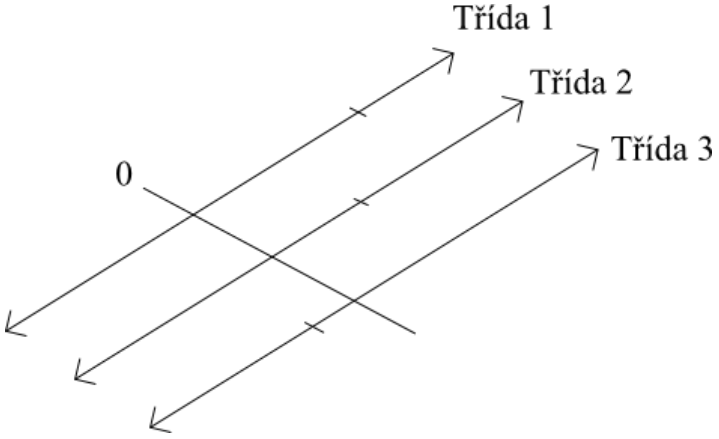
3.5 STROMOVÝ PROGRAM SIMULUJÍCÍ IMPERATIVNÍ ZÁPIS S VEKTOROVÝM VÝSTUPEM}

Velkou nevýhodou programů, které produkuje běžná gramatická evoluce je jednohodnotový výstup. Takový výstup vedl často k překrývání intervalu nebo nemožnosti vytvořit kvalitní klasifikátor. Proto byly navrženy terminály gramatiky, které dokážou napodobit chování klasických nestromových programů.

Na obrázku 3.3 je naznačen možný výstup programu, který má více než jednu výstupní hodnotu. U navrhovaných metod lze takový výstup získat, pokud použijeme pro výsledky výpočtů registry – proměnné, do kterých lze uložit výsledky výpočtů.

Dalším možným problémem byla u předchozí metody kombinace jednohodnotového výstupu a volby rozsahu intervalů až na základě výstupních hodnot programu. Jelikož měl každý jedinec jinou množinu intervalů, která nebyla kódována v chromozomu, nebylo možné jedince efektivně křížit a mutovat.

Výhoda tohoto přístupu proti předchozímu je taková, že jednotlivé výstupní hodnoty mohou mít v programech samostatné větve stromu a je možné tak programy efektivně kombinovat při křížení a mutaci.



Obrázek 3.3.: Program s vektorovým výstupem

Vyhodnocení výstupu se pak nabízí například porovnáním výstupu, což je vektor s hodnotami, s vektory představujícími bázi vektorového prostoru o *n* dimenzích. Kde *n* je počet tříd, které klasifikujeme. Jednoduše pak lze vyhodnotit vzdálenost každého vektoru k bázovým vektorům prostoru. Každý bázový vektor je svázán s třídou objektů. Nejbližší bázový vektor je pak prohlášen za výstup.

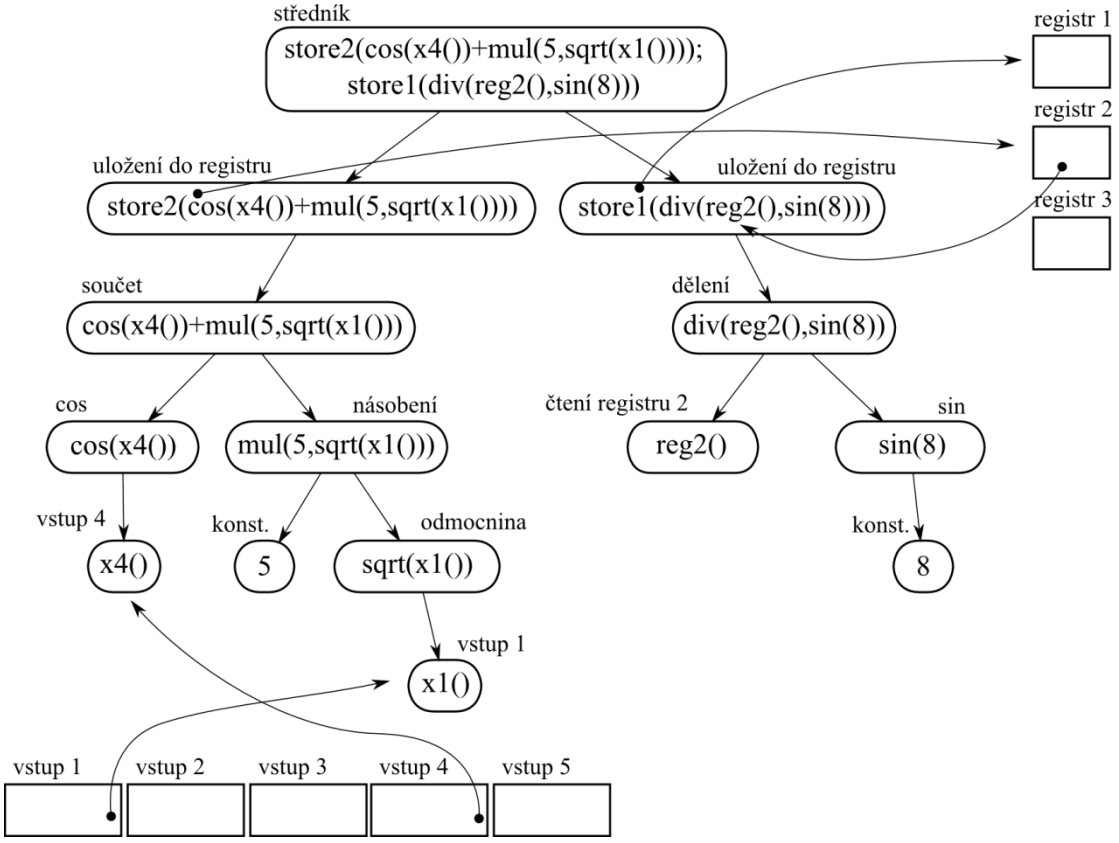
Takový výstup lze použít i k dalšímu zpracování v neuronové síti, nebo jen k transformaci obrazu, pokud bychom se pouze rozhodli, že použijeme gramatickou evoluci k vývoji obrazového filtru.

3.5.1 Registry

Terminální symboly gramatiky pro registry slouží k uchování hodnot mezivýpočtů a zároveň jsou používány k předání výsledků výpočtu programu. Realizovány jsou pomocí pole hodnot. Gramatická evoluce si může zvolit, který registr bude program modifikovat nebo číst.

3.5.2 Operátor středník

Terminál gramatiky středník umožňuje napodobit chování programů zapsaných v klasickém imperativním jazyku. Zachovává ovšem stromovou strukturu programů a tím i možnost velmi snadného křížení a mutace pomocí záměny podstromů programu.



Obrázek 3.4.: Struktura programu s uzlem středník, vektorovým vstupem a registry

Stromový graf programu používající operátor středník je na obrázku 3.4. Je zde demonstrován krátký program, který má přístup k pěti vstupním hodnotám a třem registrům. Program dokáže z registrů číst i do nich zapisovat.

I když je struktura programu stále stromová, jeho spuštění simuluje zápis programu *po řádcích*.

Modifikace operátoru křížení

S použitím středníku je možné zavést novou variantu křížení, kde se oba programy vybrané ke křížení spojí právě pomocí operátoru středník.

Tato operace je prováděna dodatečně po klasickém křížení. Na výstupu metody zodpovědné za křížení je tak o jednoho jedince více. Pravděpodobnost této operace se dá nastavit podobně jako pravděpodobnost standardního křížení nebo mutace.

Teoreticky by se takto dalo cíleně spojovat i více programů naráz. Implementace takovéhoto operátoru by ale byla složitější.

Cílem zavedení této operace bylo umožnit evoluční metodě slučovat klasifikátory. V počátku se v populaci vyskytovaly klasifikátory schopné najít jednu nebo dvě třídy objektů. Mnoho programů přitom umělo najít jiné objekty, než ostatní programy. Pomocí této operace mohly být takovéto programy snadno sloučeny.

Pokud by tato operace nebyla implementována, hrozilo by, že bude vybrán jeden klasifikátor, který umí nejlépe rozpoznat co největší množství tříd hned na začátku a jeho fenotyp nakonec převládne v celé populaci (uvíznutí v lokálním extrému). Ostatní kvalitní klasifikátory by tak mohly vymizet z populace.

3.5.3 Použití pro klasifikaci

Běh programu je stejný jako bylo popsáno v 3.4.1 nebo 3.4.2. Rozdíl je pouze ve výstupu programu. Vyhodnocení výsledků posuvného okna je opět stejné podle 3.2.4.

Opět by i v tomto případě bylo možné, rozdělit detekci objektů a klasifikaci do dvou kroků. Pro lokalizaci objektů bychom mohli jistě využít vytvořené klasifikátory popsané v sekci 3.4.

3.5.4 Výpočet funkce fitness

Pro výpočet funkce fitness byl navržen vzorec (3.3). Tento vzorec má v sobě integrován požadavek na schopnost klasifikaci co největšího počtu tříd. Tato podmínka je vyjádřena ve fitness funkci, jelikož gramatika jako taková nemusí generovat program, který by dokázal rozpoznat všechny třídy. Naopak může být vygenerován program, který například rozpozná jen některé třídy, ale velice dobře. Hodnota *SR – Success Rate* pak může být vyšší pro takový program, než pro program, který umí rozpoznat více tříd, ale na horší úrovni. Je lepší mít program, který dokáže rozpoznat více tříd a ten upravit pomocí mutace a křížení k lepším výsledkům. Druhá část vzorce pro výpočet fitness přidává na hodnocení právě takovýmto programům. Výsledek výpočtu leží v intervalu <0, 1>.

$$Fit(M) = w_1 SR + w_2 \frac{RC}{CC} \tag{3.3}$$

- w_1, w_2 = váhy
- *SR* = míra úspěšnosti – samostatný vzorec (*Success Rate*)
- *RC* = počet rozpoznaných tříd (*Recognized Count*)
- *CC* = počet tříd, které mají být rozpoznány (*Class Count*)

Váhy w_1 a w_2 byly nastaveny podle vzorců (3.4) a (3.5). Rozdělují tak číslo 1 v poměru počet klasifikovaných tříd ku jedné.

$$w_1 = \frac{CC}{CC + 1} \tag{3.4}$$

$$w_2 = \frac{1}{CC + 1} \tag{3.5}$$

Hodnota *SR* může být vypočtena různými způsoby. Byl zvolen vzorec (3.6), který jednoduše počítá průměrnou hodnotu skóre za úspěch pro všechny třídy. Nezohledňuje se tak, pokud náhodou existuje více trénovacích vzorů v jedné třídě než v ostatních. Číslo *n* odpovídá počtu tříd.

$$SR = \frac{\sum_{i=1}^{n} \frac{S_i}{MS_i}}{n} \tag{3.6}$$

- S_i = skóre získané za detekce objektů třídy *i*
- MS_i = maximální skóre získatelné za detekce objektů třídy *i*

Úspěšná detekce je, pokud je výstup klasifikátoru vyhodnocen jako objekt odpovídající třídy v trénovacím vzoru. Vzhledem k tomu, že klasifikátor pracuje jako posuvné okno, je nutné tyto odpovědi filtrovat podle vzdálenosti. Podle velikosti okna je nastavena určitá hranice.

Pro výpočet skóre S_i byl navržen jednoduchý vzorec (3.7) zohledňující vzdálenost od středu objektu *D*.

$$S_i = \sum_{points} \frac{1}{1 + D} \tag{3.7}$$

Hodnoty MS_i jsou počítány pro celý obraz stejným vzorcem, ale nebere se v úvahu výstup klasifikátoru. Vzdálenost byla vypočtena pomocí vzorce (3.8), kde s_x a s_y jsou souřadnice středu nejbližšího objektu. Pozice okna v obraze je dána vektorem *(r, c)*.

$$D = \sqrt{(s_x - c)^2 + (s_y - r)^2} \tag{3.8}$$

4 POPIS PROGRAMU A ARCHITEKTURA TESTOVACÍHO PROSTŘEDÍ

Testovací prostředí bylo vytvořeno v programovacím jazyce Java [26]. Je dostupné jako elektronická příloha této práce. Java je objektově orientovaný, staticky typovaný jazyk se syntaxí odvozenou od jazyka C. Tento programovací jazyk byl zvolen z důvodů jeho vestavěné podpory pro práci s vlákny a synchronizaci přístupu k datům z vláken.

Jako vývojové prostředí byl zvolen program Eclipse [11]. Tento je též napsán v programovacím jazyce Java. Podporuje tvorbu projektů, spouštění, ladění. Další užitečná vlastnost je možnost instalace zásuvných modulů – byl použit systém pro správu verzí SVN [1].

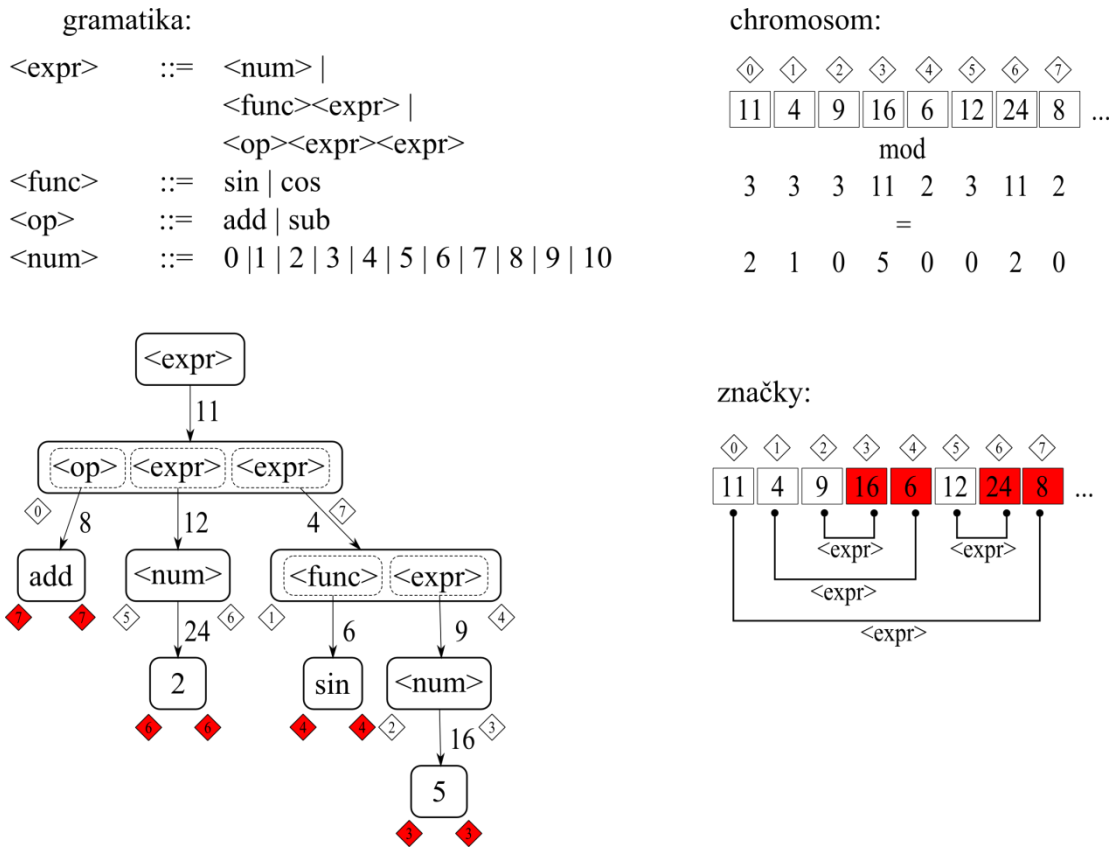
Testovací prostředí má v aktuální verzi pouze minimální uživatelské rozhraní. Úlohy se definují a konfigurují přímo ve zdrojovém kódu. Byl implementován pouze systém zobrazování grafů a obrázků pro vykreslení zadání nebo průběhu některých úloh a průběhu hodnoty fitness funkce.

Implementace evolučních metod není triviální záležitost. V této kapitole jsou popsány podrobněji implementované algoritmy. Některá inovativní řešení, která zrychlují běh celého procesu, jsou popsány podrobně.

4.1 IMPLEMENTACE PODSTATNÝCH FUNKCÍ PRO GRAMATICKOU EVOLUCI

Pro rychlejší běh byly implementovány funkce, které umožňují získat tzv. značkovací vektor a funkce pro přímé získání spustitelného stromu. V následujících kapitolách budou tyto postupy popsány.

4.1.1 Získání značkovacího vektoru



Obrázek 4.1.: Získání značkovacího vektoru při překladu chromozomu

Na obrázku 4.1 je zachycen výsledek překladu chromozomu pomocí gramatiky. Zároveň je v dolní části zobrazen graf, kde jednotlivé hrany znázorňují použití číslice z chromozomu. V malých kosočtvercích je potom udána pozice, kterou překlad dané větve začal a skončil. Tyto pozice určují hranice značek. Terminální symboly jsou vyznačeny červeně. Terminály jsou vždy vybírány posledním prvkem části chromozomu, kterou značka vymezuje.

Záměnou červeně vyznačené číslice dosáhneme pouze výběru jiného terminálu – nestrukturální mutace. Strukturální mutaci je možné provést tak, že vyměníme celý obsah značky (nové pole číslic nemusí být nutně stejně dlouhé). Je ovšem nutné zajistit, aby pole, které nahrazuje obsah značky vycházelo ze stejného neterminálního symbolu. Ke značkám je tedy nutné uložit i informaci, jaký neterminální symbol překládají.

4.1.2 Získání značkovacího vektoru přímo při překladu chromozomu

Důležitý je i postup získávání značek. Na následujících obrázcích bude popsán postup získání značkovacího vektoru. Gramatika a výsledek odpovídá obrázku 4.1.

Značky jsou získávány přímo při překladu použitím dané stromové struktury. Jako kořen stromu je vždy určen počáteční symbol gramatiky. K jeho překladu je použita první číslice chromozomu (11). Jelikož v gramatice máme 3 možnosti překladu neterminálu <*expr*>, použijeme operaci *modulo* (zbytek po dělení) a získáme 11 *mod* 3 = 2. Použijeme tedy přepis s indexem 2 (počítáme od nuly). Výsledek přepisu je tedy <*op*><*expr*><*expr*>.

Hotové značky pak jednoduše posbíráme z jednotlivých uzlů grafu a zapíšeme je do seznamu značek k danému chromozomu. S použitím takovéhoto algoritmu automaticky paralelně s tvorbou výsledného fenotypu jedince získáváme vektor značek.

4.1.3 Přímý překlad chromozomu na spustitelný strom

Dalším vylepšením, které urychluje proces gramatické evoluce je přímý překlad chromozomu. Tento krok odstraňuje nutnost použití parseru, který by převáděl textový řetězec na spustitelnou strukturu. Jakmile je při překladu vytvořen terminál, je vložen do zásobníku. Díky zpětnému překladu je jako poslední terminál vložen nejvyšší uzel spustitelného stromu. Hned na ním je jeho první argument a za ním jeho argumenty, pokud má. Každý terminál zná počet svých argumentů, takže pomocí operace *pop* (výběr ze zásobníku) snadno získáme argumenty každého uzlu a přidáme je do stromu. Tento postup je spouštěn rekurzivně pro každý prvek zásobníku. Pokud je gramatika správně nadefinována, skončí překlad tak, že zásobník je prázdný.

4.1.4 Parser

Jednoduchý parser byl samozřejmě implementován také. K jeho funkci je potřeba gramatika, podle které rozeznává terminální symboly a váže je na funkční uzly spustitelného grafu. Ten je vytvořen opět pomocí zásobníku stejně jako v 4.1.3. Parser umí zpracovat pouze prefixovou notaci ve formátu *funkce(argumenty, ...)*, kde argument může být další funkce nebo číslice.

5 TESTY METOD A VÝSLEDKY

V této kapitole budou prezentovány výsledky rozpoznávání objektů pomocí klasifikátorů vytvořených evoluční metodou. Metody byly popsané v 3 , zde se nachází pouze výsledky a nastavení použité pro tvorbu klasifikátorů.

5.1 TESTOVACÍ SCÉNA A OBJEKTY

Pro testování metod byly zvoleny objekty, které jsou vyobrazeny na obrázku 5.1. Tyto objekty byly již dříve použity v [42], kde byla ke klasifikaci použita neuronová síť.

Testovací scéna byla vytvořena podpůrným skriptem, který umisťoval a rotoval objekty náhodně do obrazové matice zvolené velikosti. Velikost testovací scény byla zvolena 320×320 pixelů.

Jelikož jde o malou sadu objektů zasazených do testovací scény, očekávala se od genetického algoritmu vysoká úspěšnost při klasifikaci. Šlo hlavně o ověření schopnosti navržených metod nalézt klasifikátor nebo detektor.

a) Třída 1 b) Třída 2 c) Třída 3 d) Třída 4 e) Třída 5

Obrázek 5.1.: Testovací objekty ve stupních šedi

5.2 KLASIFIKÁTOR S JEDNOHODNOTOVÝM VÝSTUPEM

Jako první byly testovány metody navržené a popsané v 3.4.

5.2.1 Dvoufázový přístup

Dvoufázový přístup vykazoval lepší výsledky i rychlejší průběh (díky spouštění klasifikátoru pouze nad body označenými detektorem). Důležité bylo natrénovat detektor tak, aby opravdu rozpoznal všechny objekty, pokud se to nepodařilo, nemohl už i kvalitní klasifikátor rozpoznat neoznačený objekt. Klasifikátor i detektor byly tvořeny nezávisle. Tento postup byl publikován v [21].

V tabulce 5.1 je uvedena gramatika, která byla použita k překladu chromozomů na spustitelné programy. Gramatika je postavena tak, aby produkovala programy, které dokážou získat vhodnou sekvenci maticových operací nad odpovídajícím regionem tak, aby výsledná statistická analýza byla schopná rozřadit výstupy nad objekty do nepřekrývajících se intervalů – tím je zaručena bezchybná detekce objektů.

Neterminály	Množina přepisů
<expr>	<matrix_stats><matrix><region>
<matrix>	<img_matrix_source> | <math_fun_matrix><matrix> | <math_op_matrix><matrix><matrix>
<region>	<region_generator> | <region_op><region><region>
<img_matrix_source>	s_img() | s_sobel() | s_blur()
<math_fun_matrix>	mat_pow() | mat_p_sqrt() | mat_abs()
<math_op_matrix>	mat_add() | mat_sub() | mat_mul() | mat_p_div()
<matrix_stats>	ms_avg() | ms_stdev() | ms_mode()| ms_min() | ms_max()
<region_generator>	r_center_point() | r_round_quarter() | r_round_half() | r_round_full() | r_square_quarter() | r_square_half()
<region_op>	r_and() | r_or() | r_xor()

Tabulka 5.1.: Gramatika použitá k hledání klasifikačního programu

Pomocí gramatiky zadané v tabulce 5.1 byly postupem popsaném v 3.4.2 vytvořeny programy 5.1 a 5.2. Jako první byl nad obrazem program detektoru objektů. Tento označil místa, kde byl nalezen objekt a pouze v odpovídajících místech byl spuštěn klasifikátor.

```
ms_mode(mat_pow(mat_pow(mat_pow(s_img()))),r_square_quarter())
```

Program 5.1.: Detektor.

Intervaly a nastavení velikosti posuvného okna pro program 5.1:

- Objekt: <-0.004133847245740075, 0.00780259296127764>
- Velikost posuvného okna: 7 pixelů

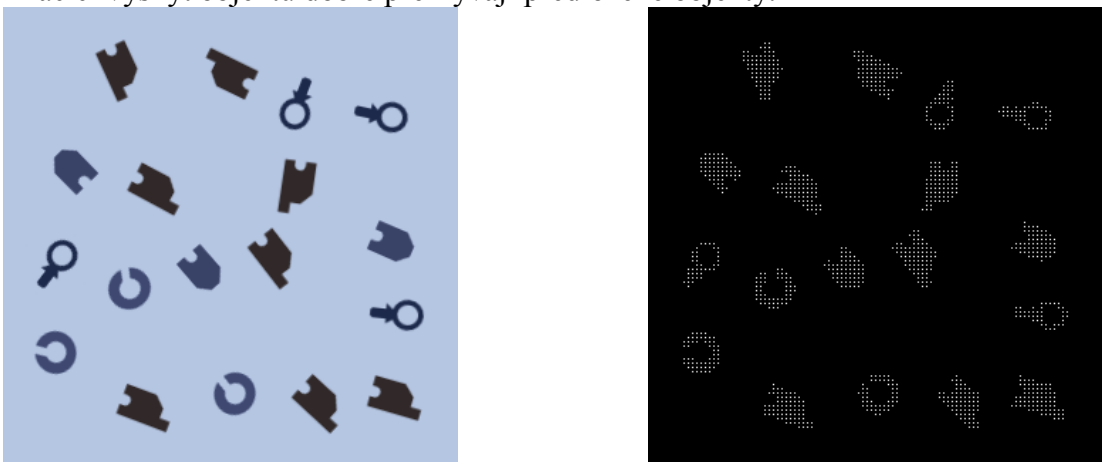
```
ms_avg(mat_p_div(mat_p_div(mat_add(s_img(),mat_pow(s_img())),mat_p
ow(mat_add(mat_pow(s_sobel()),s_blur()))),mat_pow(mat_add(mat_pow(
mat_add(mat_pow(s_img()),mat_pow(s_blur()))),mat_pow(mat_add(mat_p
ow(s_sobel()),s_blur()))))), r_round_full())
```

Program 5.2.: Klasifikátor.

Intervaly a nastavení velikosti posuvného okna pro program 5.2:

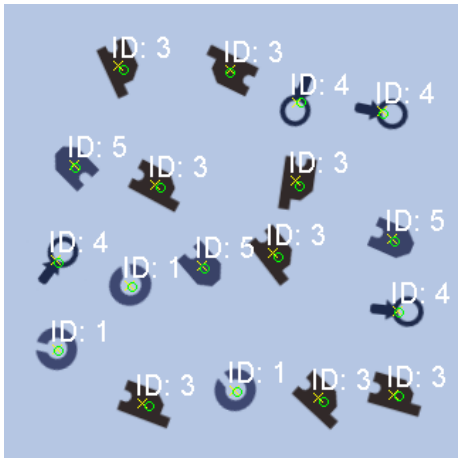
- Class 1: <40.97913746198387, 60.12324904845637>
- Class 2: <385.99089482347165, 566.6794275679941>
- Class 3: <1422.2313057348517, 2091.6064983746714>
- Class 4: <271.6726553671749, 399.10379640103235>
- Class 5: <116.74479981522084, 171.2372449892687>
- Velikost posuvného okna: 45 pixelů

Na obrázku 5.2 je výstup detektoru spuštěného jako posuvné okno s krokem 2. Je vidět, že bílé tečky, značící výskyt objektu dobře překrývají předložené objekty.



Obrázek 5.2.: Výstup detektoru

Na obrázku 5.3 je konečné vyhodnocení, které proběhlo na základě analýzy výsledků nad jednotlivými shluky bodů. Nejčastěji udaná hodnota ve shluku byla určena jako výsledek klasifikace objektů. Zelené kolečko značí těžiště objektu nalezeného detektorem a úspěch klasifikace. Žluté křížky jsou středy objektů dané v učících datech.



Obrázek 5.3.: Vyhodnocený výstup klasifikátoru

Další test, který byl proveden se zaměřil na schopnost rozpoznat mírně modifikovaný obraz objektu. Do učící množiny bylo mimo základních obrazů zaneseno i několik obrazů s přidaným náhodným šumem. V tabulce 5.2 jsou výsledky pro testovací obrazy s různými hladinami šumu.

Hodnoty šumu jsou udány jako maximální náhodná hodnota, která byla přidána ke každému pixelu. Všechny varianty vstupního obrazu měly hodnoty v intervalu <0, 1>. Takže například pro řádek tabulky, kde je udána hodnota šumu 0,04 to znamenalo, že k číslům byla přidána až hodnota 0,04.

Čas v tabulce je součtem času detektoru i klasifikátoru. Vždy je ovlivněn počtem nalezených bodů detektorem. Měřeno na procesoru Intel Core2Duo T6500.

Míra šumu	Počet objektů	Počet nalezených bodů detektorem	Úspěšnost [%]	Čas [ms]
0,00	18	1093	100,00	2030
0,02	16	921	100,00	1790
0,04	17	969	94,11	1830
0,06	18	948	38,88	1812
0,08	17	1041	0	1985

Tabulka 5.2.: Výsledky klasifikace

5.2.2 Klasifikátor využívající operátor středník

V této části práce jsou vypsány nastavení metody popsané v části 3.5 a její výsledky.

Neterminály	Množina přepisů
<expr>	<register_write> | semic(<expr>,<expr>) | if(<logic>,<expr>,<expr>
<register_write>	rWrite0(<math>) | ... | rWrite*n*(<math>)
<math>	<number> | <math_fun>(<math>) | <matrix_stats>(<matrix>) | <register_read>() | <math_op>(<math>,<math>)
<logic>	lgtn(<math>,<math>) | smtn(<math>,<math>)
<matrix>	iIm | iSob | iTh | <matrix_op>(<matrix>,<matrix>) | <matrix_op_simple>(<matrix>,<math>) | <matrix_fun>(<matrix>)
<number>	-20.0 | -19.5 | ... | 19.5 | 20.0
<math_op>	add | sub | mul | pdiv
<math_fun>	pow | sqrt | log | exp
<matrix_op>	mAdd | mSub | mMul | mDiv
<matrix_op_simple>	mAddC | mSubC | mMulC
<matrix_fun>	mPow | mSqrt | mLog | mExp
<matrix_stats>	mAvg0_3 | mAvg0_5 | mAvg0_10 | mAvg0_15 | mAvg0_20 | mAvg3_5 | mAvg5_10 | mAvg10_15 | mAvg15_20 | mSum0_3 | mSum0_5 | mSum0_10 | mSum0_15 | mSum0_20 | mSum3_5 | mSum5_10 | mSum10_15 | mSum15_20
<register_read>	rRead0 | ... | rRead*n*

Tabulka 5.3.: Gramatika použitá k hledání klasifikačního programu

```
semic(semic(rWrite4(mSum10_15(mDiv(iSob,mSub(iIm,iTh)))),rWrite2(l
og(add(mSum5_10(iSob),mSum10_15(mDiv(iSob,mSub(iIm,iTh)))))),if(l
gtn(5.5,mSum5_10(iSob)),semic(rWrite5(pow(log(mSum5_10(mDiv(iIm,mS
```

```
ub(iIm,iTh))))))),rWrite3(mSum10_15(mDiv(iSob,mDiv(iSob,mSub(iIm,iT
h))))))),rWrite1(mAvg15_20(iTh))))
```

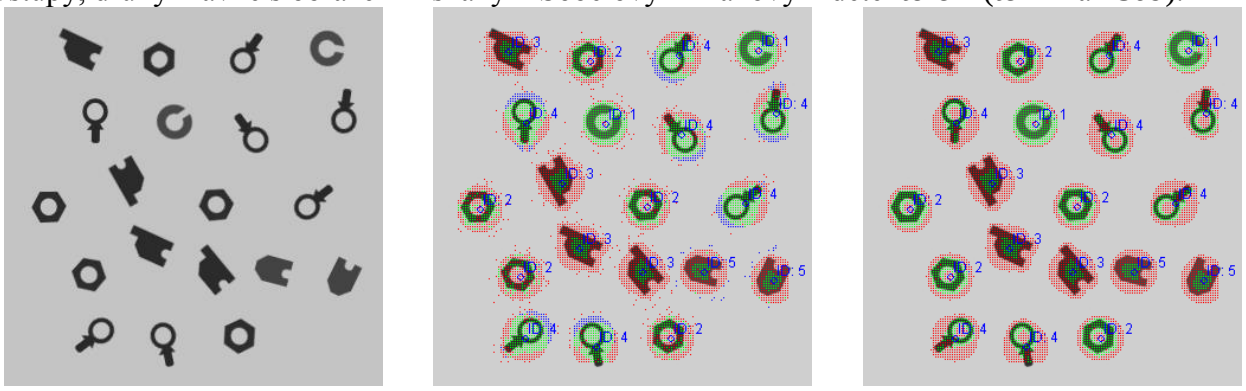
Program 5.3.: Program 1.

```
semic(if(lgtn(16.5,mSum5_10(iSob)),if(lgtn(5.5,mSum5_10(iSob)),sem
ic(if(lgtn(5.5,mSum5_10(iTh)),rWrite3(pdiv(mul(5.5,mSum15_20(iSob)
),mAvg0_20(iSob))),rWrite0(16.5)),rWrite5(10.5)),rWrite0(-7.5)),rW
rite4(mAvg0_20(iSob))),rWrite2(mAvg5_10(mExp(mDiv(mSubC(mSubC(iIm,
mSum15_20(iSob)),mSum15_20(iSob)),iSob)))))
```

Program 5.4.: Program 2

Pro ukázku jsou uvedeny 2 programy získané pomocí gramatické evoluce (program 5.3 a program 5.4). Velikost posuvného okna byla nastavena na 41 pixelů.

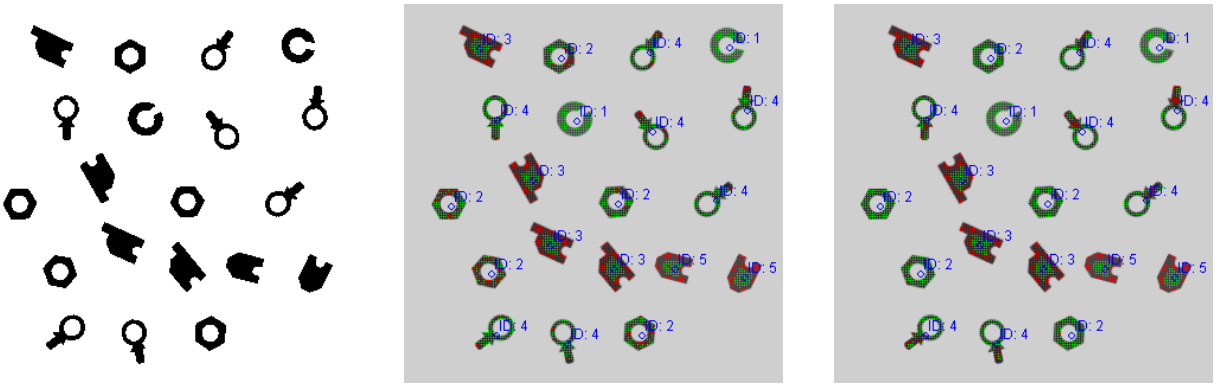
Programy jsou složitější než v předchozí sekci. První program pracuje s různými obrazovými vstupy, druhý hlavně s obrazem získaným Sobelovým hranovým detektorem (terminál *iSob*).



Obrázek 5.4.: Testovací scéna (vlevo) a výsledky detekce pro program 1 (uprostřed) a program 2 (vpravo)

Na obrázku 5.4 je výstup obou programů spuštěných nad testovací scénou. Červené tečky jsou nesprávně hlášené výstupy klasifikačního programu. Zelené tečky jsou výstupy shodující se s trénovacími daty v definované vzdálenosti (polovina posuvného okna). Modré tečky jsou správné výstupy ve větší vzdálenosti.

Je vidět, že podobně jako u předchozí verze, má jednofázový výstup hodně nesprávných identifikací v oblasti kolem objektů. Pro zlepšení výstupu bylo provedeno prahování. Klasifikátor byl spouštěn jen v místech, kde prahování nalezlo objekt, obrázek 5.5. Prahování nahradilo v tomto případě program detektoru. Na složitější scéně by ale bylo vhodné použít detektor nebo jinou metodu pro získání objektů z obrazu.



Obrázek 5.5.: Prahovaný obraz (vlevo) a výsledky detekce pro program 1 (vlevo) a program 2 (vpravo) – prahování

Na výstupech, kde bylo použito prahování je u objektů s otvorem vidět, že definovaná trénovací data, která obsahují střed bodu nejsou úplně vhodná. Jelikož střed objektu leží právě v tomto otvoru. Je to pozorovatelné hlavně u programu 1 (na obrázcích uprostřed). Ten má sice vyšší hodnotu fitness, ale objekty s identifikátorem 2 (matka) rozpoznává hůře. Vyšší hodnotu fitness má, protože má některé správné detekce blíž středu objektu (objekt 2 a objekt 4). Možná oprava by byla modifikací trénovacích dat nebo úpravou vzorce pro výpočet skóre (3.7).

Časová náročnost byla měřena jen orientačně. Program 1 potřebuje asi 0,5 sekundy a program 2 potřeboval přibližně sekundu pro ohodnocení všech míst daných prahovaným obrazem. Tedy 2238 bodů pro krok posuvného okna 2. Měřeno na procesoru Intel Core2Duo T6500.

5.3 ZHODNOCENÍ VÝSLEDKŮ

Navržené metody jsou, co se týká úspěšnosti velice dobré. Problémy nastávají, pokud klasifikátoru předložíme objekty, na které není připraven (například je obraz jinak nasvícen nebo je zašumělý). Toto se dá ovšem odstranit rozšířením množiny trénovacích vzorů a přidáním vhodných terminálů do gramatiky.

Pokud jde o rychlost rozpoznávání, hlavním faktorem, který definuje dobu trvání procesu je počet spuštění programu. S rostoucí plochou obrazů bude kvadraticky růst i počet spuštění. Přístup s detektorem nebo prahováním sice umožní běh programu zrychlit, ale na větších obrazech bude trpět stejným nedostatkem, jen ne v takové míře. Řešením je ponechat lokalizaci objektů jiné metodě, což je ostatně v praxi běžným postupem.

Přístupem rozpoznávání s posuvným oknem poměrně značně prodloužíme i čas tvorby klasifikátoru, neboť je nutné spustit všechny programy v populaci nad všemi vzory pro učení.

V porovnání s neuronovou sítí je proces učení o mnoho delší, neboť při učení neuronové sítě pracujeme jen s jedním klasifikátorem. U evolučních metod máme klasifikátorů právě tolik, kolik je členů populace.

U gramatické evoluce je vhodné spustit proces učení vícekrát, abychom získali více klasifikátorů, ze kterých vybereme ten nejlepší. Zajímavou možností je vložit vybrané kvalitní jedince do nového procesu gramatické evoluce a nechat je dále zlepšovat. Případně je možné výsledek procesu gramatické evoluce ručně doladit. To u neuronové sítě nelze.

V produkční fázi má použitý princip posuvného okna stejný vliv i na neuronovou síť. Struktura představených klasifikátorů je jednodušší než struktura neuronové sítě. Proto bude rozpoznávání objektů pomocí vyvinutých klasifikátorů rychlejší.

Díky tomu, že jsou vyvinuté klasifikátory, co se týká struktury, obvykle jednodušší než neuronové sítě, může být jednodušší jejich aplikace v tzv. *embedded* řešeních (systémy se specifickou funkčností, často realtime). Je to proto, že nejsou nutné speciální knihovny a je menší prostorová i časová náročnost výpočtu. Schopnost vyvinout program přímo v odpovídajícím programovacím jazyku je také využitelná.

Silným argumentem pro použití navržené metody je možnost získat invarianci vůči rotaci a stranovému převrácení na úrovni klasifikátoru. Tímto požadavkem se sice prodlouží učení, jelikož je zmenšena množina obrazových regionů, které můžeme použít k výpočtu statistik, nicméně v produkční fázi se odstraní fáze hledání výchozí polohy objektu.

6 ZÁVĚR

Cílem práce bylo navržení metody, která použije gramatickou evoluci k tvorbě krátkých počítačových programů, které jsou použitelné jako klasifikátory pro rozpoznávání objektů v obraze. Hlavní aplikační oblastí výsledků je průmyslová robotika, kde je požadována vysoká přesnost a rychlost rozpoznávání.

Hlavním výstupem této práce je ucelený postup rozpoznávání objektů v obrazech, založený na gramatické evoluci. Dále byly navrženy vzorce pro výpočet hodnot funkce fitness. Tyto postupy jsou použitelné i obecně mimo oblast počítačového vidění. Inovativní je i řešení používající operátor středník a registry, k získání vícehodnotového výstupu z klasifikačního programu. Tento vícehodnotový výstup se dá lépe vyhodnotit a případně může posloužit jako vstup do dalšího stupně klasifikace. Dalším hodnotným výstupem je popsaný způsob implementace algoritmu značkování a překladu chromozomu na spustitelné stromy.

Pro testování algoritmů bylo také vytvořeno programové prostředí v jazyce Java, které může dále posloužit pro výuku nebo řešení reálných problémů. V tomto prostředí není implementována jen gramatická evoluce, ale obecný mechanismus evolučních metod.

Gramatická evoluce a evoluční metody obecně nejsou nejrychlejšími metodami pro řešení optimalizačních úloh. Je to proto, že pracují s populací kandidátních řešení a jejich hromadné ohodnocování zabírá nejvíce času. Pokud tedy existuje algoritmus, který je navržen přímo pro řešení nějaké konkrétní úlohy, obvykle evoluční metodu překoná. Naopak nepřekonatelnou výhodou těchto algoritmů, je jejich schopnost hledat řešení problémů, které jsou z pohledu klasických metod neřešitelné ať už proto, že neexistuje konkrétní metoda nebo je prohledávaný prostor řešení příliš velký. Vysvětlením je, že evoluční algoritmy poskytují jakýsi obecný návod, jak tyto složité problémy řešit, aniž bychom o nich museli příliš vědět. Stačí najít způsob jak možná řešení problému zakódovat do chromozomu a ohodnotit.

Metody gramatické evoluce jdou ještě dále a oprošťují jejich uživatele od nutnosti navrhnout reprezentaci chromozomu a mechanismu křížení či mutace. Stačí jen dodat funkci pro výpočet hodnoty fitness a gramatiku. Výhodou může být i výstup ve zvoleném formátu – dáno použitou gramatikou. Navíc dokáže gramatická evoluce optimalizovat i strukturu výsledného počítačového programu.

Pro použití gramatické evoluce je tedy nejdůležitější navrhnout správně funkci pro výpočet hodnoty fitness a dát evoluční metodě takové nástroje, aby mohla zvyšovat kvalitu jedinců podle této funkce. Je důležité si uvědomit, že pokud je fitness funkce chybně nadefinována, evoluční

metoda bude optimalizovat jedince k co nejlepším výsledkům, ale ti budou k řešení úkolu nepoužitelní.

Výhoda navržených metod je v tom, že navržené klasifikátory mají obecně méně složitou strukturu než neuronové sítě. Jejich použití je proto vhodné právě v aplikacích, kde je k dispozici omezený výpočetní výkon a operační paměť.

6.1 PŘÍNOS PRÁCE

6.1.1 Přínos vědecký

Hlavním přínosem práce jsou tyto dosažené výsledky:

- Ucelený postup rozpoznávání objektů v obrazech, založený na gramatické evoluci.
- Vzorce pro výpočet fitness použitelné v klasifikačních problémech.
- Navržení terminálů středník a registr, které umožní programu vyvinutému gramatickou evolucí mít vícehodnotový výstup a zachovat snadné křížení a mutaci pomocí výměny podstromů.
- Popis efektivního algoritmu pro značkování chromozomu.
- Popis tvorby spustitelného programového stromu během překladu chromozomu.

Během řešení bylo publikováno několik článků na téma gramatická evoluce a její využití v aplikacích. Hlavním tématem bylo rozpoznávání objektů, ale gramatická evoluce byla testována i pro klasifikaci jiných dat a pro predikci časových řad: [21][22][23][41].

6.1.2 Praktický

Praktickým přínosem je vytvoření testovacího prostředí pro spouštění a vývoj metod založených nejen na gramatické evoluci a aplikace gramatické evoluce do oblasti, kde se tradičně používají jiné metody umělé inteligence.

Testovací prostředí může být využito k demonstraci funkce evolučních metod studentům. Případně použito k řešení dalších úloh.

7 SEZNAM POUŽITÉ LITERATURY

[1] APACHE FOUNDATION: *Subversion.* Online: http://subversion.apache.org/

[2] BAY, H. ESS, A., TUYTELLAARS, T.: Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*. Vol. 110, 2008, No. 3, pp. 346–359. ISSN 1077-3142.

[3] BEAUMONT, D., STEPNEY, S.: Grammatical Evolution of L-systems. In *CEC '09. IEEE Congress on Evolutionary Computation*, 2009, pp. 2446–2453, 2009. ISBN 978-1-4244-2958-5.

[4] BEBIS, G., et al.: Genetic Object Recognition Using Combinations of Views. *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 132–146, 2002, ISSN 1089-778X

[5] BHANU, B., SUNGKEE, L., DAS, S.: Adaptive image segmentation using genetic and hybrid search methods. *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 31, No. 4, pp. 1268–1291, 1995. ISSN 0018-9251.

[6] BOHÁČ, M., LÝSEK, J., et al.: Face Recognition by Means of New Algorithms. In *MENDEL 2011, 17th International Conference on Soft Computing*. Brno University of Technology, Czech Republic, pp. 324–329, 2011, ISBN 978-80-214-4302-0.

[7] BURKE, E.K. GUSTAFSON, S. KENDALL, G.: Diversity in Genetic Programming: An Analysis of Measures and Correlation With Fitness. In *IEEE Transactions on Evolutionary Computation*. Vol. 8, No. 1, pp. 47–62, 2004. ISSN 1089-778X.

[8] ČÍŽEK, L., ŠŤASTNÝ, J.: Comparison of Genetic Algorithm and Graph-based Algorithm for the TSP. In *MENDEL 2013, 19th International Conference on Soft Computing*. Brno University of Technology, Czech Republic, pp. 433–438, 2013. ISSN 1803-381, ISBN 978-80-214-4755-4.

[9] CORTES, C., VAPNIK, V. N.: Support-Vector Networks. *Machine Learning*, Vol. 20, 1995.

[10] DAE N. CHUN, HYUN S. YANG: Robust Image Segmentation Using Genetic Algorithm with a Fuzzy Measure. *Pattern Recognition*, Vol. 29, No. 7, pp. 1195–-1211, 1996. ISBN 1-59593-186-4.

[11] ECLIPSE FOUNDATION: *Eclipse*. Online: http://www.eclipse.org/

[12] GARAI, G., CHAUDHURI, B. B.: A Novel Grid Pattern Matching Technique with Hybrid Genetic Algorithm. In *TENCON 2009 - IEEE Region 10 Conference*, pp. 1–6, ISBN 978-1-4244-4546-2.

[13] GHOSH, P., MITCHELL, M.: Segmentation of Medical Images Using a Genetic Algorithm. In *GECCO '06 Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp. 1171–1178, 2006.

[14] HAN LIU, DING LIU, JING XIN: Real-time Recognition of Road Traffice Sign in Motion Image Based on Genetic Algorithm. In *Proceedings of International Conference on Machine Learning and Cybernetics*, pp. 83–86, 2002, ISBN 0-7803-7508-4.

[15] HOPCROFT, J. E., ULLMAN, J. D.: *Formal languages and their relation to automata*. Boston: Addison-Wesley, 1969, 288 p. ISBN 0-201-02983-9.

[16] HOPFIELD, J. J.: Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences of the USA*, Vol. 79, No. 8, pp. 2554–2558, 1982.

[17] KOHONEN, T.: Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, 43. 79, No. 1, pp. 59–69, 1982.

[18] KOZA, J. R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, 1992, ISBN 0-262-11170-5.

[19] KYU-PHIL HAN, KUN-WOEN SONG, EUI-YOON CHUNG, SEOK-JE CHO, YEONG-HO HA: Stereo matching using genetic algorithm with adaptive chromosomes. *Pattern Recognition*, Vol. 34, No. 9, pp. 1729–1740, 2000. ISSN 0031-3203

[20] LOWE, G. D.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 91–110, 2004, ISSN 0920-5691.

[21] LÝSEK, J., ŠŤASTNÝ, J., MOTYČKA, A.: Object Recognition by means of Evolved Detector and Classifier Program. In *MENDEL 2012, 18th International Conference of Soft Computing*, Brno University of Technology, Czech Republic, pp. 82-–87, 2012, ISSN 1803-3814, ISBN 978-80-214-4540-6.

[22] LÝSEK, J., ŠŤASTNÝ, J., MOTYČKA, A., Comparison of Neural Network and Grammatical Evolution for Time Series Prediction. In *MENDEL 2013, 19th International Conference on Soft Computing*. Brno University of Technology, Czech Republic, 2013, pp. 215-220, ISSN 1803-3814, ISBN 978-80-214-4755-4.

[23] LÝSEK, J., ŠŤASTNÝ, J., Classification of Economic Data into Multiple Classes by Means of Evolutionary Methods. In *Enterprise and Competitive Environment 2013*, Brno, Czech Republic, 2013.
[24] McCULLOCH, W., PITTS, W.: A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, pp. 115–133, 1943.
[25] OBJECT AID: *Object Aid UML Explorer*. Online: http://www.objectaid.com/
[26] ORACLE CORP.: *JAVA*. Online: http://www.java.com/
[27] ORTEGA, A., DALHOUM, A. A., ALFONSECA, M.: Grammatical evolution to design fractal curves with a given dimension. *IBM Journal of Research and Development*, Vol. 47, No. 4, pp. 483–493, 2003. ISSN 0018-8646.
[28] PEREZ, B. C., OLAGUE, G.: Evolutionary learning of local descriptor operators for object. In *GECCO '09, Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pp. 1051–1058, USA, 2009, ISBN 978-1-60558-325-9.
[29] POPELKA, O., RUKOVANSKÝ, I., OŠMERA, P.: Grammatical evolution with backward processing. In *Proceedings of Fourth internation conference on soft computing applied in computer and economic enviroments*, pp. 89-96, 2006. ISBN 80-7314-084-5.
[30] POPELKA, O. ŠŤASTNÝ, J.: *Uplatnění metod umělé inteligence v zemědělsko-ekonomických predikčních úlohách*, 48 p., MENDELU, 2009, ISBN 978-80-7375-340-5
[31] POPELKA, O., ŠŤASTNÝ, J.: WWW Portal Usage Analysis Using Genetic Algorithms. *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*, 2009, Vol. 6, pp. 201–208, ISSN 1211-8516.
[32] PROCESSING ORG.: *Processing*. Online: http://www.processing.org/
[33] PRUSINIKEWICZ, P., LINDENMAYER, A.: *The algorithmic beauty of plants*, 228 p., 1991. ISBN 0-387-97297-8.
[34] ROTH, G., LEVINE, D. M.: Geometric Primitive Extraction Using a Genetic Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 9, pp. 901–905, 1994, ISSN 0162-8828
[35] ROUPEC, J.: *Vývoj genetického algoritmu pro optimalizaci parametrů fuzzy regulátorů*, disertační práce. Brno: VUT, 2001.
[36] RUMELHART, D. E., HINTON, G. E., WILLIAMS, R. J.: Learning representations by back-propagating errors. *Nature*, Vol. 323, pp. 533–536, 1986.
[37] RYAN, C., O'NEILL, M.: *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Kluwer, 2003. 160 pp. ISBN 978-1-4020-7444-8
[38] SANG KYOON KIM, DAE WOOK KIM, HANG JOON KIM: A recognition of vehicle licence plate using a genetic algorithm based segmentation. In *Proceedings of International Conference on Image Processing*, pp. 661–664, 1996. ISBN 0-7803-3259-8.
[39] SEETHARAMAN, G. S.: A Genetic Coding Structure and Algorithms for Image Segmentation. *Biologically Structured Computing Systems*, 1997
[40] SUGANTHAN, P. N.: Structural pattern recognition using genetic algorithms. *Pattern Recognition*, Vol. 35, No. 9, pp. 1883–-1893, 2002, ISSN 0031-3203
[41] ŠKORPIL, V., LÝSEK, J., et al.: Grammatical Evolution as a Learning Process for Multi-class Object Detection. In *Recent Researches in Communications and Computers*, 16th

WSEAS International Conference on Communications, Kos, Greece, 2012, pp. 101-105, ISBN 978-1-61804-109-8.

[42] ŠŤASTNÝ, J., ŠKORPIL, V.: *Analysis of Algorithms for Radial Basis Function Neural Network*, Personal Wireless Communications, Springer, 2007, Issue 1, pp. 54–62, ISSN 1571-5736, ISBN 978-0-387-74158-1.

[43] TSANG, P. W. M.: A genetic algorithm for affine invariant recognition of object shapes from broken boundaries. *Pattern Recognition Letters*, Vol. 18, No. 7, 1997, pp. 631–-639, ISSN 0167-8655

[44] WINKLER, J. F., MANJUNATH, B. S.: Genetic Programming for Object Detection. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pp. 330–335, 1997.

[45] ZHANG, M., CIESIELSKI, V. B., ANDREAE, P.: A domain-independent window approach to multiclass object detection using genetic programming. *EURASIP Journal on Applied Signal Processing*, Vol. 2003, No. 8, pp. 841–859, 2003, ISSN 1687-6180.

[46] ZHANG, M., CIESIELSKI, V.: Genetic Programming for Multiple Class Object Detection. *Advanced Topics in Artificial Intelligence*, pp. 180–192, 1999, ISBN 978-3-540-66822-0.

8 AUTOROVO CV

Osobní údaje

Ing. Jiří Lýsek, narozen 18. srpna 1984 v Přerově

Pedagogická činnost

V rámci doktorského studia na FSI VUT: Informatika, Automatizace

V rámci pracovního poměru na MENDELU: Word, Excel, Výpočetní technika, Webové aplikace, Algoritmizace, Aplikační programové vybavení, Aplikace vývojových technik

Přehled zaměstnání

2013-doposud:	Mendelova univerzita v Brně, Akademický pracovník – asistent
2012-2013:	Mendelova univerzita v Brně, Technický pracovník pro výuku
2008-doposud:	Programátor PHP

Vzdělání a akademická kvalifikace

2008-2009:	Magisterské studium, VUT Brno, Fakulta strojního inženýrství, Obor aplikovaná informatika a řízení, titul Ing.
2004-2008:	Bakalářské studium, VUT v Brno, Fakulta strojního inženýrství, Obor strojní inženýrství, titul Bc.
1999-2003:	SPŠ Přerov Havlíčkova 2, obor strojírenství

9 ABSTRACT

This thesis deals with usage of evolutionary methods, grammatical evolution particularly in application for object recognition in an image. Basic principles of object recognition and evolutionary methods with focus on grammatical evolution are described. The core of the thesis lies in design of techniques and methods for classifier programs creation using grammatical evolution. Also the designed fitness formula is presented. In the end, created testing and development environment in Java programming language is described.