TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

SYSTÉM PRO AUTOMATICKÉ DETEKOVÁNÍ A ROZPOZNÁVÁNÍ ROSTLIN

Diplomová práce

Studijní program:	N2612 – Elektrotechnika a informatika
Studijní obor:	1802T007 – Informační technologie
Autor práce:	**Bc. Matěj Kolář**
Vedoucí práce:	doc. Ing. Josef Chaloupka, Ph.D.

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2015/2016**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Matěj Kolář**
Osobní číslo:	**M13000193**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Systém pro automatické detekování a rozpoznávání rostlin**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Seznamte se s problematikou rozpoznávání obrazu a s knihovnou OpenCV.
2. Vytvořte, případně stáhněte databázi obrazů, ve kterých se nacházejí květy, případně listy rostlin.
3. Navrhněte a naprogramujte algoritmy vhodné pro detekci a rozpoznávání rostlin.
4. Otestujte navržené algoritmy na databázi obrazů s rostlinami.
5. Vytvořte webový systém - online atlas rostlin, ve kterém si bude moci uživatel rozpoznat vlastní obrazy rostlin. Tento systém by měl dále umožňovat archivovat obrazy rostlin a informace k těmto rostlinám.

Rozsah grafických prací:	**Dle potřeby dokumentace**
Rozsah pracovní zprávy:	**cca 40 až 50 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **Davies, E., R.: Machine Vision - Theory, Algorithms, Practicalities. Morgan Kaufmann Press. UK, 2005, ISBN 0-12-206093-8**

[2] **Hlaváč V., Sedláček M.: Zpracování signálu a obrazu, Skripta FEL ČVUT, Praha 2000, ISBN 80-01-02114-9**

[3] **Šonka, M., Hlaváč, V., Boyle, R.: Image Processing, Analysis, and Machine Vision. PWS Publishing, USA, 1999, ISBN 0-534-95393-X**

Vedoucí diplomové práce:	**doc. Ing. Josef Chaloupka, Ph.D.** Ústav informačních technologií a elektroniky
Konzultant diplomové práce:	**Ing. Karel Paleček** Ústav informačních technologií a elektroniky
Datum zadání diplomové práce:	**14. září 2015**
Termín odevzdání diplomové práce:	**16. května 2016**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -
L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
vedoucí ústavu

V Liberci dne 14. září 2015

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 16. 5. 2016

Podpis: 

Poděkování

Tímto bych chtěl poděkovat vedoucímu diplomové práce panu doc. Ing. Josefu Chaloupkovi, Ph.D. za užitečné konzultace, rady a věcné připomínky k této práci. Dále mu také děkuji za jeho trpělivost a vstřícnost. Také bych rád poděkoval konzultantovi Ing. Karlu Palečkovi za jeho rady týkající se klasifikace a klasterizace.

Abstrakt

V rámci této diplomové práce byl vytvořen systém pro automatické rozpoznání rostlin. První část systému se zabývá detekcí rostlin v obraze, segmentací a klasifikací. Pro detekci a segmentaci je použita metoda GrabCut. Příznakové vektory obrazů rostlin jsou generovány pomocí metody HOG (histogramy orientovaných gradientů) pro popis tvaru a metodou klasterizace k-means v barevném prostoru HSV pro popis barvy. Systém dále využívá metody PCA (analýza hlavních komponent) pro redukci dimenze příznakových vektorů. Druhá část sytému je vytvoření webového portálu pro správu rostlin. Ten je vytvořen pomocí NodeJS, HTML a Bootstrap.

Klíčová slova: rozpoznání rostlin, databáze rostlin, OpenCV, HOG, k-means, PCA, NodeJS

Abstract

In this thesis is created a system for automatic detection and recognition of plants. The first part of the system deals with a plant detection in image, segmentation and classification. For the detection and segmentation is used the GrabCut method. Feature vectors of images of plants are generated using the method HOG (histograms of oriented gradients) for description of the shape and for description of the color the method k-means clusterization in HSV color space is used. The system uses PCA (principal component analysis) method to reduce the dimension of feature vectors. The second part of the system is to create a web portal for managing the plants. It is created using Node.JS, HTML and Bootstrap.

Keywords: flower recognition, flower database, OpenCV, HOG, k-means, PCA, NodeJS

Obsah

Seznam tabulek

Seznam ilustrací

Seznam použitých zkratek

HOG – histogramy orientovaných gradientů

RGB – barevný prostor obsahující složky R (red), G (green), B (blue)

HSV – barevný prostor obsahující složky H (hue), S (saturation), V (value)

CMY – barevný prostor obsahující složky C (cyan), M (magenta), Y (yellow)

Gray – šedotónový barevný prostor

BSD – Berkeley Software Distribution je typ licence pro svobodný software

SRM – Statistical Region Merging

GMM – Gaussian Mixture Model

MRF – Markov Random Field

PCA – Principal Component Analysis

LARK – Locally Adaptive Regression Kernels

SVM – Support Vector Machines

NoSQL – typ nerelační databáze (non SQL)

JSON – JavaScript Object Notation

HTML – značkovací jazyk pro tvorbu webových stránek

CSS – kaskádové styly pro úpravu vzhledu

C++ – programovací jazyk

Úvod

V oblasti počítačového vidění je z pohledu člověka velice atraktivní jakýkoli systém pro automatické rozpoznávání objektů v obraze. V dnešní době existuje více takových systémů, jejichž funkcí je například rozpoznávání lidí, aut, rostlin či stromů. Aby byl člověk schopný tyto předměty rozpoznat a určit je potřeba, aby se nejdříve naučil jakým způsobem má přemýšlet a na co se zaměřit. Stejné je to i s počítačovými systémy. Nejprve je nutné určit, jak má počítač vstupní obraz zpracovat a na co se zaměřit. Jejich hlavními výhodami jsou zejména rychlost a možnost učit se na prakticky neomezeně velké databázi dat. Jedním z nich je i systém pro automatickou detekci a rozpoznání rostlin v obraze.

Takto vyhraněný systém může nalézt uplatnění hlavně u lidí, kteří nejsou zdatnými botaniky, ale chtějí se dozvědět, co jim roste na zahradě. Pro použití systému pak stačí jen vyfotografovat rostlinu a nechat systém určit, co se na ní nachází. Úspěšnost těchto systémů závisí hlavně na použité fotografii a na velikosti databáze použité pro trénování. Sofistikovanější systémy používají fotografie rostlin, které obsahují více různých pohledů dané rostliny. Rostlinu je tak možné rozpoznat podle květu, listu, stonku nebo třeba i fotografie celé rostliny. Pro tento způsob rozpoznávání je nutné vytvořit velmi obsáhlou databázi, která obsahuje každou rostlinu ze všech těchto pohledů.

Cílem této diplomové práce je vytvoření systému umožňujícího rozpoznávání rostlin a online webové aplikace pro správu fotografií k jednotlivým rostlinám. První část se zabývá vytvořením rozpoznávacího algoritmu na základě příznakových vektorů z jednotlivých fotografií databáze. Druhá část popisuje vytvoření webového systému pro správu rostlin a jejich obrazů. V poslední části je popsáno otestování algoritmu a nalezení optimálních parametrů klasifikátoru.

1 Využité metody zpracování obrazu a klasifikace

Zpracování obrazu je obor v informatice, který se zabývá zpracováním obrazového (digitálního) signálu. Vstupem je obraz reprezentovaný dvourozměrným signálem a výstupem je opět obraz nebo popis jeho charakteristiky. Základní operace zpracování obrazu jsou: transformace obrazu zvětšením nebo rotací, transformace barev do různých barevných prostorů (RGB, HSV, šedotónové barvy). Mezi základní charakteristiky patří popis obrazu pomocí histogramu, pomocí funkcí popisujících významné regiony a jejich tvar nebo pokročilé metody jako například HOG (histogramy orientovaných gradientů). Pro práci s obrazem existují dnes obsáhlé knihovny, které obsahují výše uvedené funkce a práce s obrazem je tak díky těmto knihovnám výrazně jednodušší. Jednou z těchto knihoven je i knihovna OpenCV.

1.1 Knihovna OpenCV

Je to multiplatformní knihovna pro práci s obrazem [1], vydávána s licencí BSD a s otevřeným kódem psaným v jazyce C++. V případě použití knihovny v jiném jazyce než C++ je zapotřebí dodatečné obalové knihovny, která zpřístupní všechny funkce přímo v konkrétním programovacím jazyce. Výhodou této funkcionality je zachovaní rychlosti výpočtů nativních funkcí knihovny i v jiných jazycích než C++.

Prvky knihovny OpenCV jsou:

- modul jádra – definuje základní datové struktury a funkce používané dalšími částmi knihovny,
- zpracování obrazu – obsahuje lineární a nelineární filtrování obrazu, geometrické transformace, konverze barevných prostorů, výpočty histogramů, detekce rysů a objektů a další funkce,
- další prvky zahrnující práci s videem, 3D obrazem, 2D detektory a deskriptory, uživatelské rozhraní pro zobrazení obrazu a algoritmy akcelerované pomocí grafické karty.

1.2 Segmentace obrazu

Segmentace obrazu je metoda zabývající se zpracováním obrazu a nalezením oblastí, které jsou si podobné nebo spolu nějak souvisí a jsou z hlediska informační

hodnoty relevantní. Typickým příkladem segmentace je nalezení a ohraničení popředí obrazu od pozadí. Segmentace se využívá hlavně při automatickém zpracování obrazových dat z různých oborů – lékařství, počítačové vidění, extrakce informací, rozpoznávání a identifikace [2].

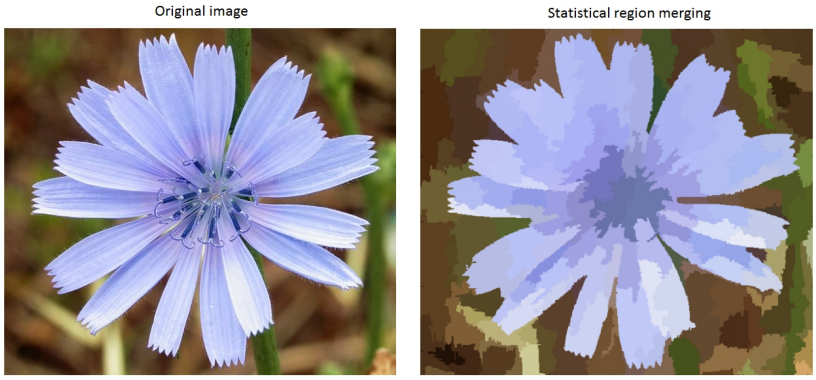
Základní metody segmentace:

- Prahování – jedná se o nejjednodušší způsob segmentace, která rozdělí obraz do dvou částí popředí a pozadí. Rozdělení využívá hodnot jasu pixelů, kdy pixely s hodnotou nižší než práh jsou označeny jako pozadí a ostatní jako popředí. Tato metoda nelze použít tam, kde je práh rozhodování nejednoznačný nebo obraz obsahuje velké množství šumu.
- Regionální metody – jsou založené na podobnosti jednotlivých pixelů či regionů, jako podobnost může být použit jas pixelu, statistická vlastnost či jiná relevantní hodnota (hodnota složky hue z barevného prostoru HSV). Průběh algoritmu metody je založen na slučování či rozrůstání regionů se stejnými nebo podobnými hodnotami dané vlastnosti. Algoritmus pro slučování regionů nejprve rozdělí obraz do regionů tak, že každému pixelu přiřadí jiný region. Poté prochází obraz a slučuje jednotlivé regiony na základě podobnosti. Algoritmus končí v případě, že již nelze žádné další regiony spojit. Druhým typem je algoritmus pro rozrůstání regionů, který pro inicializaci používá semínka, což jsou jednotlivé pixely patřící do daných regionů. Poté jsou tyto regiony iterativně rozšiřovány, dokud daný region rozšířit lze. V opačném případě algoritmus končí.
- Hranové metody – využívají pro detekci oblasti, ve kterých dochází k velké změně jasů pixelů. Na základě velikosti změny je vytvořen nový obraz se zvýrazněnými hranami, který je ještě nutno dále upravit. V nově vzniklém obraze mohou být díky šumu i hrany, které hranami nejsou. K odstranění těchto hran se používá prahování s rozhodovací funkcí volenou ke konkrétním obrazům.

Pokročilé metody segmentace:

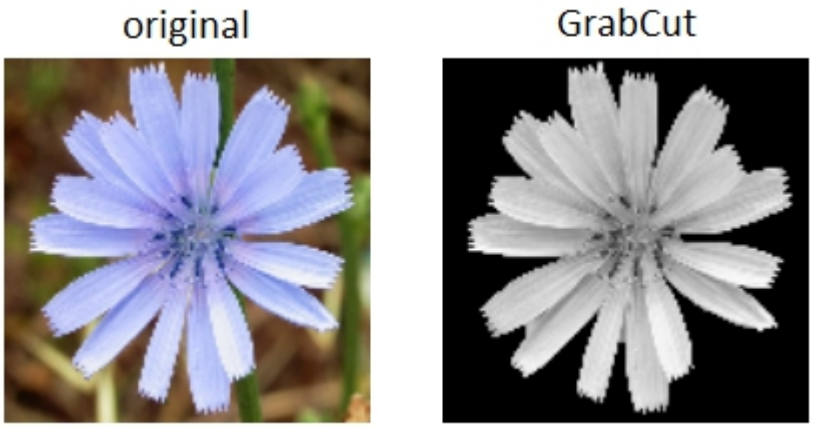
- SRM – Statistical Region Merging je druh regionální metody segmentace obrazu, která využívá slučování pixelů do regionů na základě jejich vzdálenosti

a velikosti rozhodovacího intervalu. Je vhodná zejména k redukci regionů v obraze, kdy jednotlivé regiony mohou reprezentovat barvy v obraze. Algoritmus na základě vstupních parametrů (cílový počet regionů, minimální a maximální velikost regionů) dokáže zredukovat počet regionů a výstupem je například obraz s určitým množstvím barev [3].



Obr. 1: Ukázka algoritmu SRM na květině

- GrabCut – vychází ze starší metody s názvem GraphCut a byla popsána v roce 2012 Carstenem Rotherem, Vladimirem Kolmogorovem a Andrewem Blakem na univerzitě Cambridge [4]. Je to metoda segmentace založená na iterovaných řezech grafem. Inicializace algoritmu se provádí jako specifikace popředí pomocí ohraničujícího čtyřúhelníku. Nejprve je odhadnuto rozdělení barev popředí a pozadí pomocí GMM (Gaussian Mixture Model), to je pak použito pro sestrojení MRF (Markov Random Field) pro všechny pixely, které následně označí na základě optimalizace funkce energie s upřednostněním regionů se stejným označením. Poté je spuštěna optimalizace založená na řezu grafem k získání hodnoty těchto regionů. Tento proces je opakován, dokud není dosaženo konvergence. Výsledný obraz lze ještě dodatečně upřesnit uživatelem, který zpřesní výběr regionů pozadí a popředí.



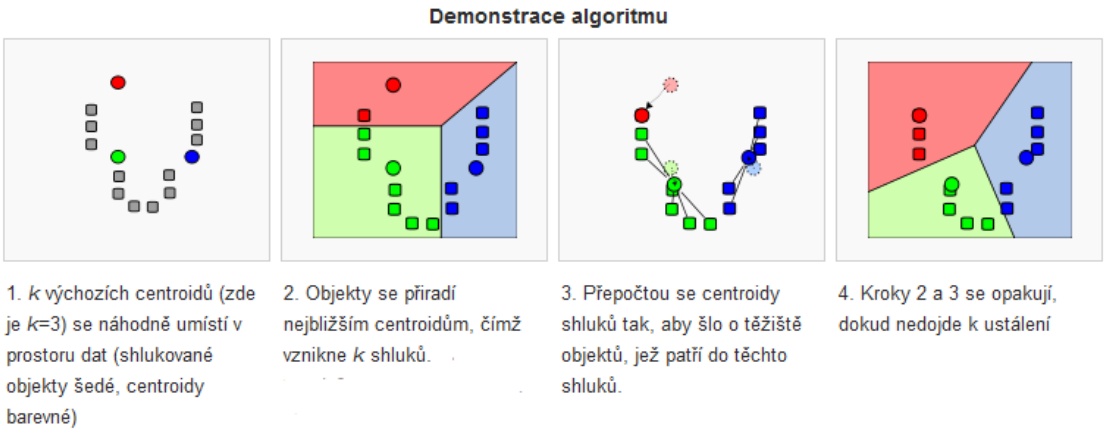
Obr. 2: Ukázka algoritmu GrabCut na květině

1.3 Klasterizace

Klasterizace je ve statistice úloha, která seskupuje data tak, že data ve stejné skupině mají větší podobnost než tato data vzhledem k jiné skupině. Hlavní využití klasterizace nalezneme ve strojovém vidění, rozpoznávání objektů, obrazové analýze nebo při kompresi dat. Složitost klasterizace není pouze o hledání vhodného algoritmu, ale také o hledání vhodných parametrů klasterizačního algoritmu. Mezi základní parametry patří vzdálenostní funkce, hustota a počet klasterů (jader). Některé algoritmy potřebují pro svůj běh trénovací data, na kterých se přizpůsobí konkrétním datům a také data pro vyhodnocení, která jsou klasterizována podle natrénovaných hodnot [2].

Klasterizační algoritmy:

- založené na těžišti – k-means je algoritmus, který rozděluje prostor do k různých shluků, určených vlastními centroidy. Začíná tak, že nejprve náhodně určí počáteční pozice k centroidů a poté přiřadí všechny body prostoru k tomu centroidu, ke kterému má daný bod nejblíže. Poté spočítá těžiště jednotlivých shluků bodů a na to umístí nový centroid. V dalším kroku jsou znovu spočítány vzdálenosti bodů *k* centroidům a přiřazeny k nejbližšímu z nich a tak dále, dokud se poloha centroidů mění. Algoritmus končí v případě, že se pozice žádného centroidu již dále nemění.

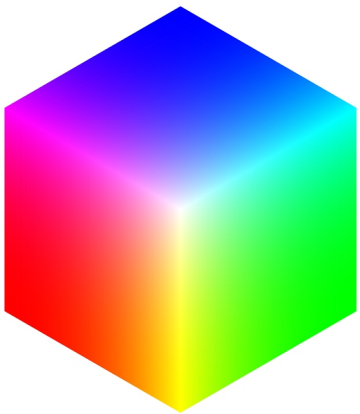


Obr. 3: Algoritmus k-means

1.4 Barevné modely – RGB, HSV, Gray

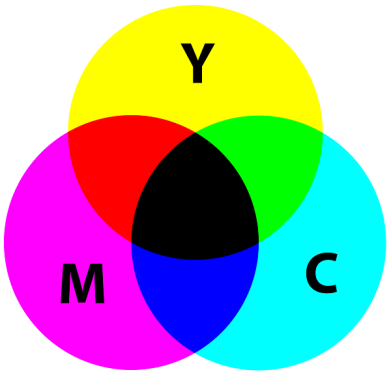
Barevný model definuje základní barvy, ze kterých se ostatní barvy míchají (skládají). Dále se snaží reprezentovat skutečné barvy co možná nejvěrněji podobné reprezentaci pomocí lidského oka. Využití jednotlivých modelů se odvíjí od konkrétní úlohy, kde chceme používat nějakou reprezentaci barvy.

RGB model je základní aditivní model využívající 3 barvy, na které je lidské oko citlivé – červenou, zelenou a modrou. Další barvy jsou pak tvořeny pomocí sytosti těchto barev. Model RGB může být dále rozšířen o alfa kanál, který udává průhlednost výsledné barvy.



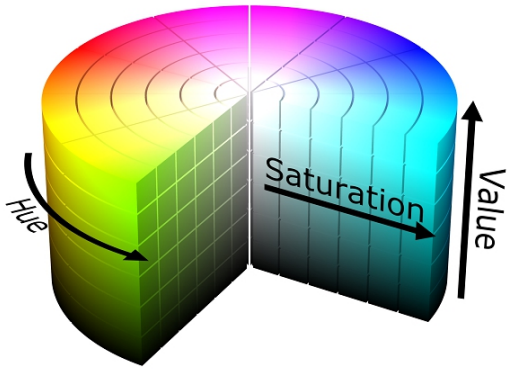
Obr. 4: Barevný model RGB

CMY je rozdílový barevný model, který se používá hlavně při tisku barev, jelikož výsledná barva je určena smícháním pigmentů jednotlivých barev.



Obr. 5: Barevný model CMY

HSV je druh RGB barevného modelu, který používá pro reprezentaci barev válcovou soustavu souřadnic. Je to model, který nejvíce odpovídá lidskému vnímání barev. Skládá se ze složek H – hue, S – saturation, V – value. Hue reprezentuje barvu na standardním barevném kole a nabývá hodnot od 0° až po 360°. Saturation je sytost barvy, kdy nejnižší sytost je bílá barva a s přibývající sytostí se mění na barvu určenou složkou hue. Value je jas nebo také tmavost a značí množství černé barvy. Tento barevný model je velice oblíbený ve strojovém vidění, segmentaci obrazu a rozpoznávání. Je odolný vůči šumu a různým světelným podmínkám [5].



Obr. 6: Barevný model HSV

1.5 Metody popisující tvary v obraze

PCA – analýza hlavních komponent (Principal Component Analysis) je transformace, která slouží k dekorelaci dat. Je to užitečná technika, která se používá

v počítačovém vidění a nalezne využití při rozpoznávání, kompresi obrazu nebo s její pomocí lze vyhledávat data či vektory ve vysokodimenzních prostorech [6]. Výpočet metody spočívá v nalezení nového prostoru nazývaného *eigenspace*, který je vypočten na základě trénovacích vektorů. Druhým krokem bývá převod trénovacích vektorů do tohoto prostoru. Testovací vektory nejsou pro výpočet potřeba a lze je převést až při klasifikaci.

Pro výpočet prostoru *eigenspace* metodou PCA je potřeba soubor *n* trénovacích vektorů o velikosti 128 * 128 pixelů v šedotónovém barevném prostoru. Jednotlivé vektory jsou převedeny na řádkové o velikosti 16384 pixelů a je z nich vytvořena matice Wp o velikosti *n* * 16384. Následně je vypočítán průměrný vektor *mu* ze všech vzorků podle vzorce:

$$mu = \frac{\sum_{i=1}^{n} \mathrm{Wp}_i}{n}$$

Takto vypočítaný vektor je odečten od všech řádků matice Wp, a tím vznikne nová matice W. Dále je spočítána kovarianční matice:

$$C = W \times W^T$$

Kovarianční matice má rozměry *n* * *n a* následně jsou z ní spočítány vlastní čísla a jím náležející vlastní vektory. Z vlastních vektorů je spočítána matice Ep tak, že jsou seřazeny dle náležících vlastních čísel od největšího po nejmenší. Posledním krokem výpočtu je vytvoření matice vlastního prostoru *eigenspace,* danou vztahem:

$$E = W^T \times Ep$$

Nyní je možné provést redukci prostoru vybráním *n* komponent matice *E*. Počet komponent matice *E* závisí na počtu vstupních vektorů, pokud jich je 1000, pak má matice *E* 1000 komponent. Vybráním prvních *n* komponent dojde ke ztrátě části informace, avšak ke značnému snížení velikosti matice *E* a tedy následnému zvýšení rychlosti výpočtu. Omezený prostor je poté použit pro převod vstupních vektorů na matici *P*:

$$P = W \times E$$

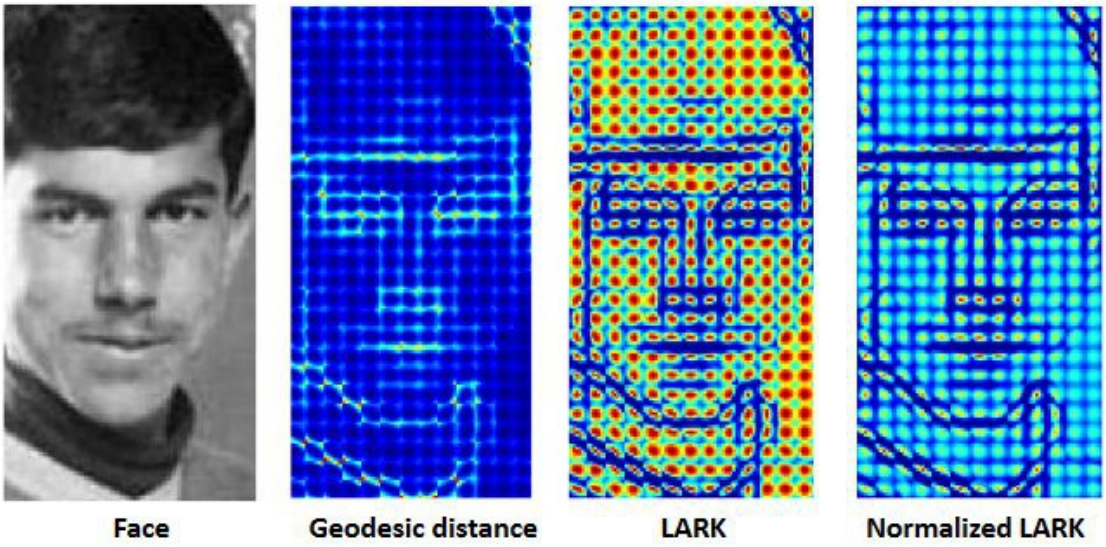
Aby bylo možné provést rozpoznávání, je nutné převést i testovací (neznámé) vektory do tohoto prostoru. To je provedeno obdobně a je k tomu potřeba dvou prvků –

průměrného vektoru *mu* a matice *E*. Nejprve je od neznámého vektoru *t* odečten průměrný vektor *mu* a poté je převeden do prostoru *eigenspace* podle vzorce:

$$Pt = t \times E$$

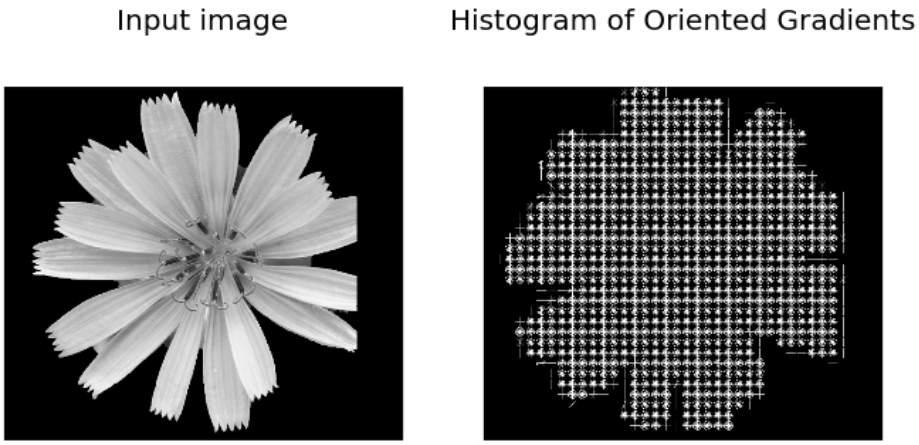
Takto převedené trénovací a testovací vektory je možné použít pro klasifikaci. Ta je provedena pomocí metody nejbližšího vzorku s pomocí některé z metrik nebo je využita jiná klasifikační metoda, například SVM.

LARK – Locally Adaptive Regression Kernels je metoda popisu obrazu, která využívá gradientů lokálních geometrických struktur. Při porovnávání podobnosti dvou pixelů je spočítána prostorová vzdálenost a jasová vzdálenost. [7]



Obr. 7: Ukázka metody LARK na obraze (převzato z [7])

HOG – histogramy orientovaných gradientů je metoda pro popis tvaru objektu v obraze. Nejprve je obraz rozdělen na stejné části pomocí mřížky. V každé této části jsou poté spočítány gradienty na základě pixelů a je vytvořen histogram z těchto gradientů. Výsledný vektor, který popisuje tvar objektu v obraze vznikne spojením všech histogramů z jednotlivých částí. Výsledný vektor slouží jako popis obrazu a objektů v něm. Jeho nevýhodou je velikost oproti vstupním datům. Pro obrázek 128*128 pixelů je výsledný HOG přibližně dvojnásobně velký. Kvůli tomu je vhodné použít metodu pro redukci dimenze například PCA.[8]



Obr. 8: Ukázka algoritmu HOG na květině

1.6 Klasifikace dat

Klasifikace dat se zabývá řešením problému zatřídění a určení kategorie neznámého vzorku. Pro správnou funkčnost klasifikátoru ve strojovém vidění je nejprve nutné klasifikátor natrénovat pomocí trénovacích dat, která obsahují jak informace o daném objektu, tak i informace o tom, do jaké třídy patří. Toto trénování je označováno jako učení pod dohledem. To znamená, že se klasifikátor učí na správně klasifikovaných datech. V opačném případě, kdy klasifikátor nemá k dispozici správně zatřízená data se jedná o učení bez dohledu, které se využívá například pro zatřízení podobnosti objektů.

Klasifikace vzdálenostní funkcí – je druh klasifikace s učením pod dohledem. Jedná se o jednoduchý klasifikátor využívající vzdálenostní funkci k nalezení nejbližšího vzorku se známou třídou. Neznámý vzorek je tedy klasifikován stejně jako nejbližší vzorek se známou třídou. Pro výpočet vzdálenosti je možné použít více vzdálenostních funkcí. Jednou z nich je eukleidovská vzdálenost, což je vzdálenostní funkce, která určuje jakým způsobem bude vypočítána vzdálenost mezi dvěma objekty. Pro vzdálenost dvou vektorů X a Y je definována jako:

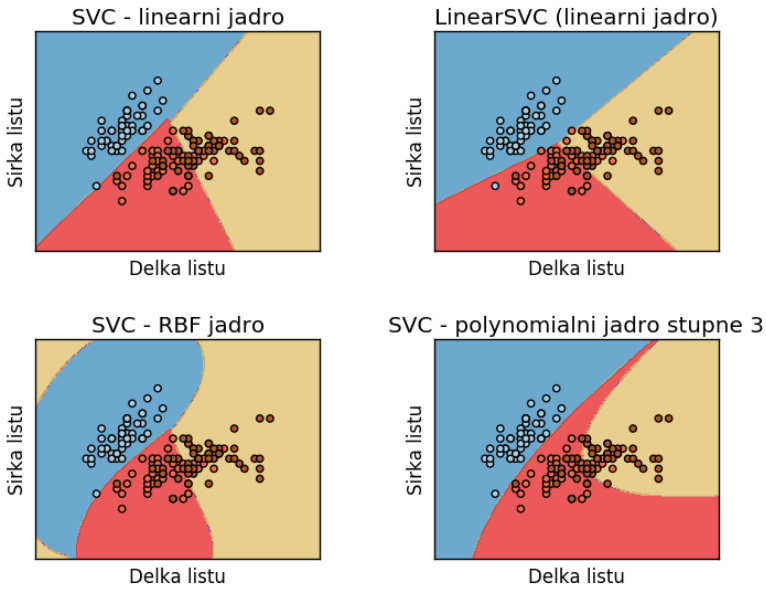
$$d(x, y) = \|x - y\| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Využití nalezne i při klasifikaci, kdy s její pomocí lze spočítat nejbližší vektor z množiny trénovacích (známých) vektorů k nějakému neznámému vektoru.

Další vzdálenostní funkcí je kosinová podobnost, která využívá funkci kosinus pro výpočet orientace vektoru a výsledkem je podobnost vektorů založená na směru vektoru a ne na velikosti. Její výpočet je dán vztahem:

$$\frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

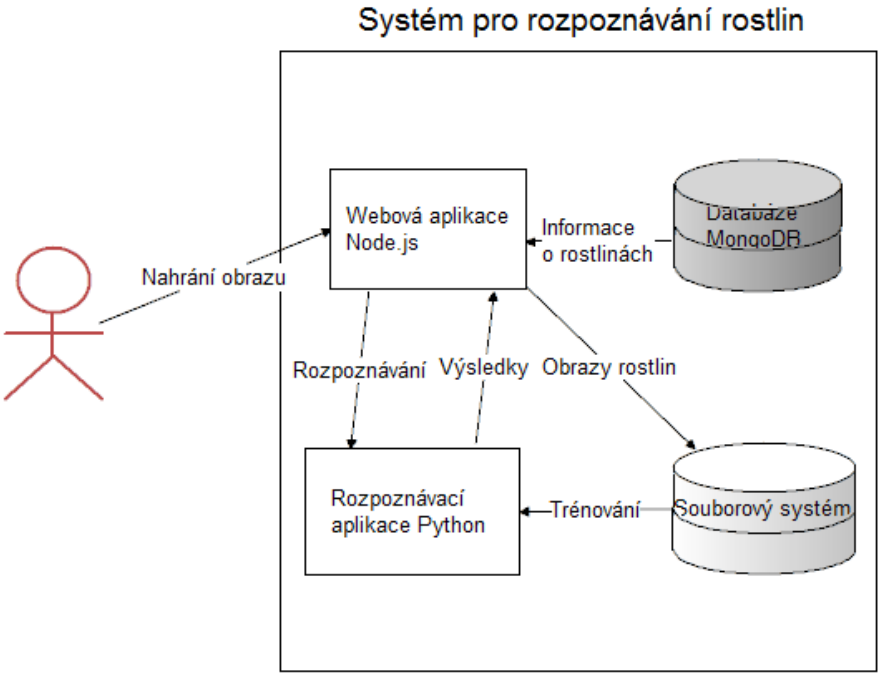
Klasifikátor SVM (Support Vector Machines) je binární klasifikátor, který pro klasifikaci využívá rozdělující nadrovinu. Tuto nadrovinu hledá tak, aby optimálně rozdělovala trénovací data a ležela uprostřed prostoru mezi dvěma shluky bodů různých tříd a měla minimální vzdálenost k nejbližšímu bodu co největší. Některá data ovšem nelze rozdělit v prostoru, ve kterém jsou do klasifikátoru zadána. Proto klasifikátor SVM převede tato data do jiného prostoru s větší dimenzí a pokusí se najít rozdělující nadrovinu v tomto prostoru. Převod na vyšší dimenzi provádí jádrová transformace, ukázku výsledného rozdělení dat s různými typy jader viz. Obr. 8. [2]



Obr. 9: Klasifikátor SVM s různými typy jádra

2 Návrh systému

Systém navržený v rámci této diplomové práce je systém pro automatickou detekci a rozpoznání rostlin v obraze. Jeho funkcí je rozpoznání rostliny z obrazu nahraného uživatelem pomocí webové aplikace. Tento obraz systém nahraje na úložiště a předá parametry do rozpoznávací aplikace, která vyhodnotí výsledek rozpoznávání. Výsledek rozpoznávání je pak prezentován uživateli jako odkaz na rostlinu v databázi, která byla vyhodnocena jako nejlepší shoda. Systém se tedy skládá z webové aplikace, databáze rostlin a rozpoznávací aplikace.



Obr. 10: Schéma navrženého systému

2.1 Webová aplikace

Webová aplikace byla vytvořena jako uživatelské rozhraní pro možnosti nahrání souboru s obrazem a rozpoznávání. Dále je zde využito databáze pro ukládání informací o rostlině.

2.1.1 Systém Node.js

Jako webový server byl zvolen softwarový systém Node.js. Je to multiplatformní software s otevřeným kódem primárně určený pro vývoj serverových webových aplikací. Vyznačuje se vysokou škálovatelností a programy, které jsou spouštěny

pomocí Node.js, jsou psány v jazyce JavaScript. Systém využívá model událostí a asynchronních operací, jako prostředí pro běh používá jádro JavaScript engine Google V8. Celá aplikace je spuštěna vždy jen v jednom vlákně a používá neblokující vstupně-výstupní volání, což ji dovoluje realizovat velké množství souběžných spojení bez nutnosti změny kontextu. Jako balíčkovací program byl vytvořen systém npm[1], který umožňuje instalovat moduly třetích stran přímo z repozitáře npm.

2.1.2 Databáze MongoDB

Webová aplikace byla navržena tak, aby bylo možné uchovávat informace o rostlinách a uživatelích. Jako databáze byla zvolena MongoDB [9]. Je to multiplatformní, dokumentově orientovaná databáze, která nepoužívá klasické relační schéma, což ji řadí do databází typu NoSQL. Pro uchovávání dat používá strukturu dokumentu, podobnou formátu JSON. Takový dokument je vytvářen dynamicky a umožňuje vytváření a integraci dat rychleji a jednodušeji.

Struktura databáze uživatelů obsahuje jméno uživatele a heslo, aby bylo možné ověřit totožnost. Ověření uživatele je nezbytné pro zpřístupnění systému.

Pro uchování informací o rostlinách byla vytvořena databáze „flowers“, která obsahuje základní informace o rostlině:

- český název,
- latinský název,
- popis rostliny,
- adresář s obrazy,
- identifikátor rostliny.

2.1.3 Knihovny Node.js

Systém Node.js je možné díky balíčkovacímu systému npm rozšířit o funkcionalitu vzhledem k početným modulům třetích stran. Základní funkcionalita systému tak může být rozšířena například o systém šablon, dynamické trasování stránek, pokročilou autorizaci a autentizaci uživatelů nebo knihovny, které umožňují sekvenční vykonávání asynchronních metod.

1 npm – balíčkovací program systému Node.js

Použité moduly systému Node.js:

Express.js je minimalistický framework využívaný hlavně jako základní kostra webové aplikace, která se stará o vykonávání jednotlivých částí kódu nad každým přijatým požadavkem. Dále se používá pro trasování dokumentů na webové aplikaci a zpřístupnění statických souborů uživateli. Tyto soubory jsou většinou soubory obsahující javascriptové a css kódy nebo obrazy. Zpracovává všechny dotazy GET a POST a volá příslušné metody pro vykonání kódů náležících jednotlivým adresám.

Jade.js je systém šablon, který umožňuje zjednodušený zápis HTML kódu pomocí značek systému jade. Systémů šablon je více typů, tento využívá šablony vytvořené a zkompilované při spuštění celé aplikace a poté jsou tyto šablony doplněny o data v závislosti na konkrétním požadavku.

Multer je knihovna pro práci se soubory, která se stará o převod formulářových dat odeslaných jako „multipart/form-data“ na soubor, který následně uloží na přednastavené umístění. Další funkcí je převod textových polí, která jsou přidružena formuláři se souborem, na objekty jazyka javascript pro použití v aplikaci Node.js.

Monk slouží pro snadnou práci s databází, přidává drobnou funkcionalitu ve formě zjednodušení dotazů do databáze. Jeho hlavní přínos je díky metodě *findAndModify*, která umožňuje uživateli provést aktualizaci záznamu v databázi.

Passport.js je střední vrstva pro Node.js, která implementuje autentifikaci a přidává možnosti pro samotné ověření uživatele a zajištění trvalého přihlášení uživatele. Jako ověření uživatele lze zvolit jakoukoli možnost, v této aplikaci bylo zvoleno ověření pomocí uživatelského jména a hesla z databáze. Po úspěšném ověření je vytvořena *session* a do ní uložen otisk aktuálně přihlášeného uživatele.

Synchronize.js umožňuje sekvenční běh asynchronních volání metod a jejich zpětných volání. Tato knihovna nalezne uplatnění všude tam, kde je potřeba počkat na provedení nějaké metody a nelze provést asynchronní kód nebo by byl takto provedený kód těžko zpracovatelný. Výsledkem je čistý sekvenční kód, který je díky této knihovně i v Node.js přehledný.

ZeroRPC.js je framework speciálně vytvořený pro jazyk Python a Node.js. Umožňuje komunikaci skrze frontu zpráv mezi jednotlivými procesy na straně serveru. Je postaven na technologiích ZeroMQ a MessagePack. Umožnuje mimo jiné i přenos

proudů dat a výjimek mezi procesy. Ošetřuje také situace zamrznutí kódu, kdy je po uplynutí stanoveného času, kdy daný proces neodpovídá, vyvolána výjimka.

2.1.4 Knihovny frontendu

Obsah zobrazený uživateli generuje aplikace pomocí systému šablon, avšak tento systém pouze vytvoří HTML kód a grafické úpravy je nutné nastylovat pomocí kaskádových stylů (CSS).

K tomuto účelu byla použita knihovna Bootstrap, která obsahuje velké množství hotových předpisů pro zobrazení prvků na webové stránce. Používá se velice jednoduše a to přiřazením příslušných tříd zobrazení k prvkům na stránce. Takto napsaná aplikace může vypadat moderně, i když tvůrce není grafický expert.

Další knihovnou pro uživatelské prostředí je knihovna jQuery, která usnadňuje práci s celým HTML dokumentem, a to zejména při procházení prvků dokumentu, práci s událostmi, animacemi a asynchronními voláními. Je napsaná v jazyku javascript a distribuuje se jako javascriptový soubor vkládaný do každé webové stránky systému.

Pro ořez obrazu uživatelem byl vybrán skript Cropper, který je postaven na jQuery a umožňuje uživateli oříznout obraz a odeslat již oříznutou část k dalšímu zpracování.

2.2 Databáze

Pro uložení informací o rostlinách byla vybrána databáze MongoDB, viz kapitola 2.1.2 a pro uložení obrazů rostlin je použit standardní souborový systém. Ten umožňuje jednodušší zálohu a obnovu nebo přidání dalších fotografií k rostlinám. Načítání obrazů k rostlinám na webové aplikaci i v rozpoznávací aplikaci probíhá vždy z příslušného adresáře rostliny. Ten je určen latinským názvem rostliny a měl by být unikátní ke každé rostlině. Pro již uživatelem oříznuté obrazy je každé rostlině vytvořen další adresář s názvem „support_img“, ve kterém jsou uloženy všechny oříznuté fotografie a slouží především ke zpřesnění květu rostliny.

2.2.1 Webová databáze

Databáze používaná na webovém systému byla stažena z webových stránek [10], kde bylo využito jak textových popisů rostlin, tak i obrazů k nim. Celý web obsahuje více

než 850 různých rostlin a průměrně 5 fotografií ke každé rostlině. Takto stažená databáze je nahrána na webovém serveru aplikace a trénování i rozpoznávání probíhá pomocí těchto fotografií.

Obr. 11: Ukázka obrazů z webové databáze - pampeliška lékařská

2.2.2 Testovací databáze

Jako testovací databáze byla použita sada 17 rostlin [11] vytvořená na Oxfordské univerzitě, obsahující nejběžněji se vyskytující druhy rostlin ve Velké Británii. Ke každé rostlině je zde k dispozici 80 fotografií květů nebo celých rostlin. Jsou zde snímky ve velkém měřítku s různým úhlem záběru a různými světelnými podmínkami. Obrazy ze stejné třídy se mohou velmi lišit a též se vizuálně více přibližovat k jiným třídám než k vlastní. Díky tomu je možné důkladně otestovat navržený algoritmus.

Obr. 12: Ukázka obrazů testovací databáze - slunečnice

2.3 Algoritmus pro rozpoznávání

Jako programovací jazyk pro aplikaci umožňující rozpoznání obrazu byl vybrán jazyk Python. Tento jazyk je přímo podporován knihovnou OpenCV, která je zručný nástroj pro práci s obrazem a existuje pro něj mnoho názorných ukázek s kódem. Dále pro tento jazyk existuje mnoho knihoven zabývajících se strojovým viděním nebo matematickými operacemi. Další podmínkou pro výběr programovacího jazyka byla možnost komunikovat s procesem, ve kterém je spuštěna webová aplikace psaná v Node.js a pro tuto potřebu je použit framework ZeroRPC. Aplikace by měla být schopna otevřít a zpracovat obraz ze souboru, vytvořit příznakové vektory z obrazu, provést natrénování klasifikátoru a poté neznámá data automaticky klasifikovat a předat výsledky aplikaci Node.js.

2.3.1 Příznaky vektorů

Byly vytvořeny dva typy příznakových vektorů – popis tvaru a barvy. V aplikaci jsou reprezentovány jako samostatné vektory, které je možné spojit za sebe tak, aby vytvořily jeden vektor, a ten pak může být použit jako příznak obou složek.

Popis tvaru je vytvořen následovně:

- načtení obrazu ze souboru a zmenšení na velikost 128*128 pixelů,
- provedení metody GrabCut pro vysegmentování květu rostliny z obrazu,
- konverze segmentované části do šedotónového barevného prostoru,
- spočítání HOG (histogramů orientovaných gradientů),
- redukce dimenze pomocí metody PCA.

Vytvoření popisu barvy:

- načtení obrazu ze souboru a zmenšení na velikost 128*128 pixelů,
- provedení metody GrabCut pro vysegmentování květu rostliny z obrazu,
- vymaskování segmentované části a převedení do HSV barevného prostoru,
- inicializace a výpočet klasterů algoritmem k-means,
- vytvoření histogramů barev.

Takto vzniklé vektory pak slouží jako příznakové vektory jednotlivých tříd pro trénování klasifikačního algoritmu. Lze z nich tedy natrénovat klasifikátor pro tvar květiny, barvu květiny nebo obojí.

2.3.2 Klasifikace příznakových vektorů

Poslední částí rozpoznávacího algoritmu je klasifikace, ta se dělí na dvě části – trénování a vyhodnocení. Nejprve jsou do klasifikátoru vložena data obsahující příznakové vektory spolu s třídou, která určuje zařazení vektoru. Po natrénování konkrétního prostoru určeného klasifikační funkcí je možné začít rozpoznávat neznámé vektory. To se děje tak, že nejprve převedeme neznámý vektor do prostoru natrénovaného v první fázi a poté na základě typu klasifikátoru klasifikujeme.

3 Rozpoznávací aplikace

Systém rozpoznávání je v této aplikaci rozdělen do menších dílčích úkonů tak, aby dohromady vytvářely ucelený systém schopný provést rozpoznávání. V první části bylo potřeba vytvořit algoritmus pro segmentaci obrazu, konkrétně pro extrakci popředí květiny. Tato oblast nese hlavní informaci o tvaru a barvě květiny, která je pro celé rozpoznávání klíčová. V další části pak již lze s takto extrahovanou informací pracovat, dále ji převádět a transformovat. V tomto momentu se algoritmus dělí na dvě nezávislé části, jednou je spočítání příznakového vektoru tvaru rostliny, druhý pak počítá příznakový vektor barvy květu. V poslední části jsou tyto dva vektory vloženy do klasifikátoru a je natrénován prostor příznaků. Ten je uložen a později je použit pro rozpoznávání neznámých vektorů předaných do tohoto algoritmu z webové aplikace.

3.1 Segmentace obrazu

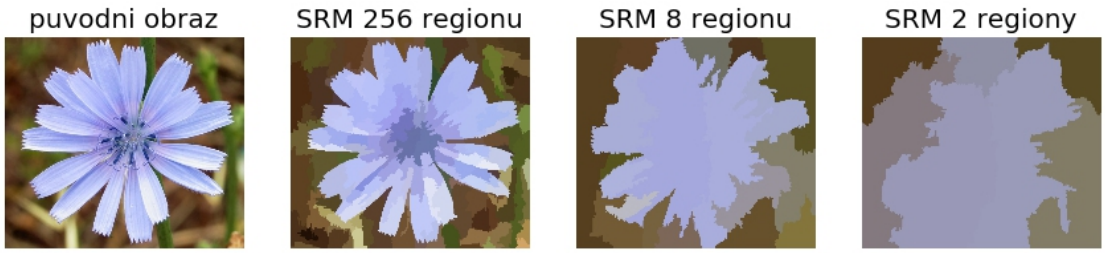
Segmentace obrazu řeší úlohu separace popředí a pozadí rostliny. Byly zde použity dva různé algoritmy a porovnány jejich výsledky.

3.1.1 SRM – Statistical Region Merging

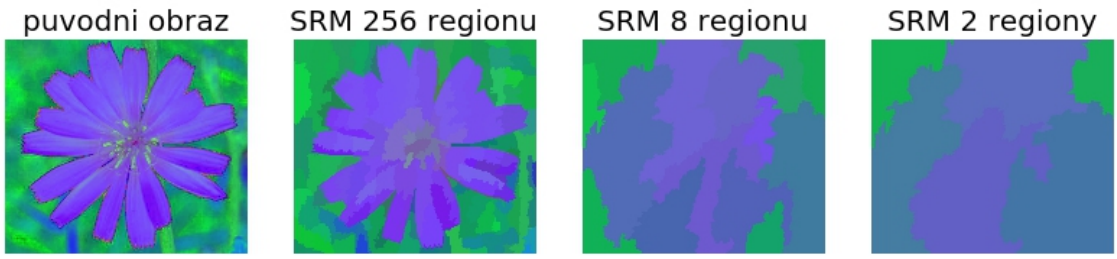
Pro segmentaci obrazu byla nejprve využita metoda SRM – Statistical Region Merging, což je metoda regionální obrazové segmentace. Použitý algoritmus byl naprogramován Olivierem Schwanderem a vydán pod licencí BSD. Bohužel tento algoritmus není schopen automaticky segmentovat popředí tak, aby označil květ rostliny jako popředí a ostatní části obrazu jako pozadí. Pro použití algoritmu byla vyzkoušena různá nastavení, avšak nastavit lze pouze přibližný počet regionů ve výsledném segmentovaném obraze. Běžně používané nastavení je kolem 256 výsledných regionů, což je spíše klasterování než segmentace. Pro použití v aplikaci bylo vyzkoušeno nastavení s různým počtem regionů. Použití metody SRM v kódu je velice jednoduché:

- importování knihovny SRM (soubor srm.py),
- předpřipravení obrazu – načtení, konverze barvy,
- spuštění algoritmu,
- normalizace výsledků pomocí podělení matice obrazu hodnotou 256.

Takto provedený algoritmus je vždy aplikován na obraz s květinou v barevném prostoru RGB. Algoritmus byl vyzkoušen i v barevném prostoru HSV, avšak výsledkem byly skoro totožné obrazy po skončení algoritmu.



Obr. 13: Ukázka algoritmu SRM v prostoru RGB



Obr. 14: Ukázka algoritmu SRM v prostoru HSV

Pro extrakci popředí byl tento algoritmus vyhodnocen jako nedostatečný a byly hledány další metody pro segmentaci.

3.1.2 GrabCut

Je metoda pro segmentaci obrazu na základě uživatelem definovaném čtyřúhelníku. Zde je využito implementace této metody pomocí knihovny OpenCV. Metoda se skládá z dílčích částí:

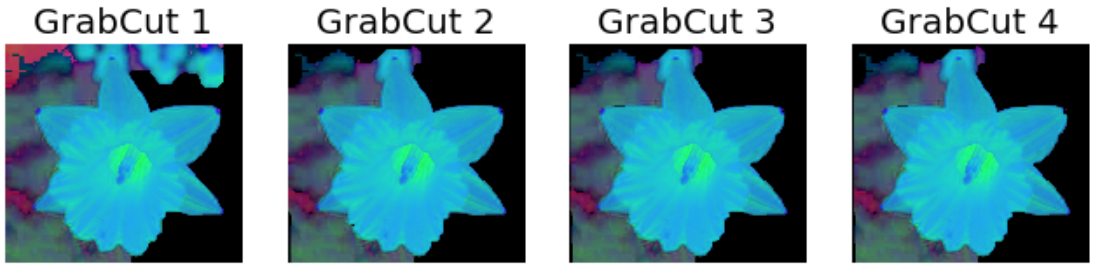
- načtení obrazu a vytvoření prázdné masky dle velikosti,
- vytvoření modelu popředí a pozadí,
- vytvoření obdélníku, který určuje popředí a pozadí obrazu – zde bylo použito více variant a byly zhodnoceny výsledky segmentace,
- provedení metody GrabCut s různým počtem iterací – každá iterace více zpřesní odhad popředí, avšak může i více oříznout popředí,
- metoda cv2.grabCut() vrací masku obrazu, kdy pixely označené čísly 0 nebo 2 jsou pixely pozadí a ostatní jsou pixely popředí,

- dalším krokem je převedení masky do formátu, kdy popředí je označeno číslem 1 a pozadí je označeno číslem 0,
- posledním krokem je vynásobení původního obrazu s touto maskou pomocí násobení matic po prvcích.

Takto vzniklý obraz má barvy původního obrazu, avšak je tu pouze vyextrahován region popředí a ostatní pixely mají hodnotu 0.

Zde bylo nutné určit hodnotu uživatelského vstupu. Jelikož není možné uživatelem zvolit hodnotu u každého obrazu, bylo nutné provést experimentální určení parametru na základě úspěšnosti při rozpoznávání. Různé výsledky segmentace s různým počtem iterací je možné vidět na Obr. 13.

Obr. 15: Ukázka metody GrabCut v prostoru RGB



Obr. 16: Ukázka metody GrabCut v prostoru HSV

Na obrazech je možné vidět nedostatečnou segmentaci v prvních dvou vzorcích v prostoru RGB, v posledním vzorku je pak vidět příliš velká segmentace, kdy chybí část květu. V barevném prostoru HSV je metoda GrabCut méně účinná než v prostoru RGB a proto nebyl tento prostor využíván.

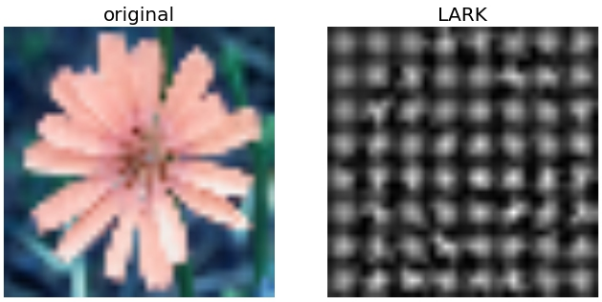
Takto vysegmentovaná část rostliny je ideální pro vytvoření jakýchkoli příznaků.

3.2 Vytvoření příznaků

Jako příznaky rostlin byly v této práci použity příznaky tvaru a barvy [12]. Jednotlivé příznaky jsou vytvořeny nezávisle na sobě ze stejné rostliny, avšak z odlišných dat. V případě tvaru je nejprve provedena segmentace metodou GrabCut, poté je obraz převeden do šedotónového barevného prostoru a je na něj aplikována metoda HOG – histogramy orientovaných gradientů. Takto vytvořený vektor je obvykle velice obsáhlý a potřebuje další úpravu v podobě redukce dimenze. Té je věnována další kapitola. Druhým příznakem vytvářeným z obrazu rostliny je příznak barvy květu. Ten je vytvořen opět z vysegmentovaného květu, kde je provedena klasterizace barev pomocí metody k-means na 128 barev a je vytvořen histogram. Takto vytvořený vektor již není nutné dále upravovat a může být použit přímo ke klasifikaci rostliny.

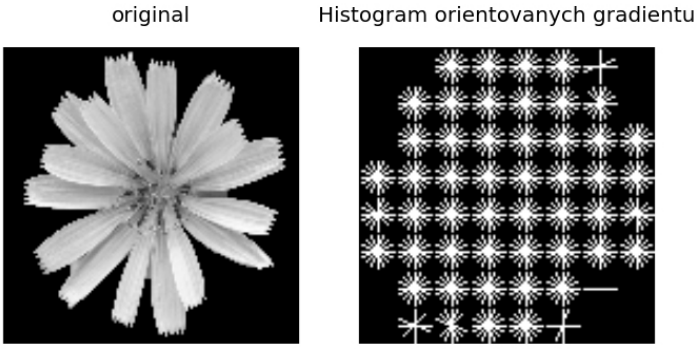
3.2.1 Tvar květu

Pro popis tvaru květu rostliny byly postupně vybrány a otestovány tři různé metody, z nich byla poté vybrána nejlepší metoda na základě rozpoznávacího výsledku a to metoda HOG. První metodou pro vytvoření příznakového vektoru tvaru bylo vytvoření řádkového vektoru pomocí jednotlivých pixelů obrazu. Zdrojový obraz byl nejprve zmenšen na výšku a šířku 128 pixelů a poté jednotlivé řady poskládány za sebe, tak aby vytvořily jeden vektor. Ten byl poté využit pro klasifikaci, avšak rozpoznávací výsledky byly nevyhovující. Další testovanou metodou byla LARK, která byla vytvořena jako pro verifikaci osob na základě obličejů. Pro tuto metodu je nutné nejprve obraz převést do odstínů šedi a poté je možné na obraz aplikovat metodu LARK. Vstupní obraz o velikostí 128*128 pixelů je touto metodou převeden na vektor o velikosti 40*40 pixelů a ten je možné použít přímo pro klasifikaci.



*Obr. 17: Ukázka algoritmu LARK o velikosti 40*40 pixelů*

Poslední testovanou metodou a také nejúspěšnější je metoda HOG. Ta je součástí knihovny OpenCV a pro výpočet vektoru HOG je nutné nejprve vytvořit objekt třídy *HOGDescriptor*, který je součástí jmenného prostoru cv2 a poté lze provést výpočet tohoto deskriptoru vložením šedotónového obrazu. Výsledkem je přibližně trojnásobná velikost oproti vstupnímu vektoru.

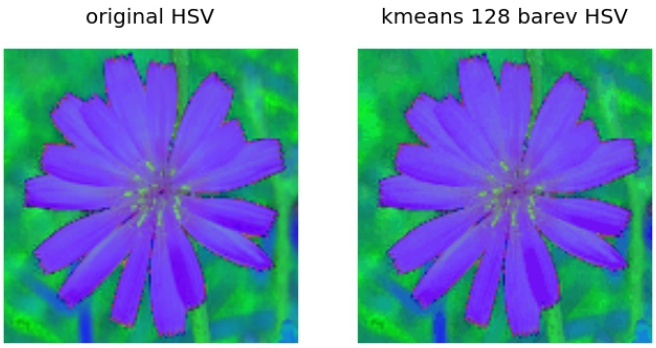


Obr. 18: Ukázka algoritmu HOG

3.2.2 Barva květu

Pro popis barvy květu je použit histogram redukovaných barev v prostoru HSV pomocí klasterizace. Tato metoda redukuje úplnou barevnou škálu obrazu, což je 24 bitů (přes 16 mil. barev) do 128 barev vypočítaných algoritmem k-means, bez vizuální ztráty informace viz. Obr. 19. Díky tomu je možné vytvořit histogram se 128 koši, který nese informaci o počtu a druhu barev vyskytujících se v obraze. Postup vytvoření histogramu:

- načtení obrazu ze souboru a zmenšení na 128*128 pixelů,
- konverze do barevného prostoru HSV,
- vytvoření řádkového vektoru o velikosti 16384*3 (počet pixelů obrazu * počet složek prostoru HSV),
- natrénování algoritmu k-means,
- predikce pomocí k-means,
- spočítání histogramu a jeho normalizace.



Obr. 19: Redukce barev pomocí k-means na 128 barev

3.3 Klasifikace

Jak již bylo zmíněno, příznakové vektory pro klasifikaci jsou dvou různých typů – příznak tvaru a barvy. Tato aplikace se dělí na 2 různé rozpoznávací skupiny – rozpoznávání rostlin a stromů. Pro účely rozpoznávání rostlin byly testovány 3 různé kombinace příznaků:

- tvar,
- barva,
- tvar ve spojení s barvou.

Pro klasifikaci stromů byl použit pouze příznak tvaru, jelikož příznak barvy není vhodný ke klasifikaci stromů na základě listů nebo plodů.

V této práci byly vyzkoušeny dvě metody klasifikace – PCA s hodnocením metodou nejbližšího vzorku a PCA v kombinaci s klasifikátorem SVM. Metoda SVM byla použita z knihovny *sklearn* pro Python.

3.3.1 PCA

Metoda PCA byla v této práci zvolena jak pro jednoduchou klasifikaci, tak především pro redukci dimenze a následné použití s binárním klasifikátorem SVM. Tato metoda umožňuje zmenšit velikost příznakových vektorů, ale zároveň zachovat značnou část původní informace. Pro klasifikaci a rozpoznávání bylo použito nastavení pro získání prvních 300 hlavních komponent, což znamená redukci až o 99% při použití HOG (46000 hodnot).

Postup výpočtu redukovaných vektorů pomocí metody PCA:

- načtení obrazů a vytvoření příznakových vektorů,
- výpočet PCA – nalezení vektorů *eigenspace* a vektoru průměrné rostliny,
- výběr určitého počtu vektorů *eigenspace* (v této práci prvních 300),
- normalizace všech vstupních vektorů odečtením průměrné rostliny,
- maticové násobení trénovacích vektorů s maticí *eigenspace* pro převod do prostoru vypočteném metodou PCA.

Takto transformované vektory jsou dále používány přímo pro klasifikaci pomocí metody nejbližšího vzorku nebo mohou být vloženy do klasifikátoru SVM.

3.3.2 Metoda nejbližšího vzorku

Touto metodou jsou v aplikaci klasifikovány vektory převedené do prostoru vypočítaném pomocí metody PCA. Ke každému vektoru trénovací sady je přiřazena informace o třídě, do které vektor patří. Na základě toho lze vypočítat vzdálenosti neznámého vektoru ke všem trénovacím vektorům. Pro výpočet eukleidovské vzdálenosti je použita funkce *linalg.norm* z knihovny numpy a pro výpočet kosinové vzdálenosti funkce *scipy_spatial.distance.cosine* z knihovny scipy. Z těchto vzdáleností je vybrán trénovací vzorek s nejnižší hodnotou a jeho třída je následně deklarována jako třída neznámého vzorku.

3.3.3 Support Vector Machine

Druhým použitým klasifikátorem pro určení třídy neznámého vzorku byl použit SVM, který je součástí knihovny scikit. Pro použití objektu SVM je zapotřebí trénovací sada vektorů, která obsahuje řádkové příznakové vektory a vektor tříd. Nejprve je nutné vytvořit objekt klasifikátoru pomocí konstruktoru SVC. Dále specifikovat povinné parametry jádra, kterými jsou parametr C, druh jádra, maximální počet iterací algoritmu a nepovinný parametr *random_state*, který zaručí stejný výsledek při každém spuštění trénování. Parametr C je volen experimentálně a jeho hodnota závisí na typu příznakových vektorů a druhu jádra. Jeho změnou dochází k velice razantní změně úspěšnosti klasifikátoru. Druh jádra byl testován také experimentálně, vždy ale vyšlo nejlépe jádro lineární. Po načtení obrazů rostlin a vytvoření příznakových vektorů jsou

tyto vektory vloženy do klasifikátoru SVM a je provedeno natrénování pomocí metody *SVM.fit.* Pro predikci výsledných tříd je nutné připravit testovací vektory obdobně jako trénovací. Do klasifikátoru jsou ale vloženy jen testovací vektory a vektor tříd je použit pro vyhodnocení úspěšnosti. Výpočet výsledných tříd je možné provést metodou *SVM.predict*, která vrací vždy nejlépe vyhovující třídu pro každý testovací vektor. Tak je možné zjistit pouze první nejlepší shodu.

Pro výpočet posloupnosti nejlepších shod ke každému vektoru byla vytvořena metoda *evaluateDecisionFunction*, která umožňuje vyhodnotit vektor rozhodující funkce. Tento vektor je možné získat pomocí metody *SVM.decision_function*, která vrací vektor rozhodujících příznaků pro každou třídu. Ten obsahuje informace o zařazení vektoru do jednotlivých tříd na základě binárního rozdělení. Jsou v něm tedy všechny kombinace dvojic tříd a jejich výsledky, zda testovaný vektor patří do první nebo do druhé třídy. Poté je nutné vyhodnotit počet kladných zařazení ke každé třídě a třída s nejvíce kladnými zařazeními je vybrána jako třída výsledná. Takto vyhodnocená rozhodovací funkce je dále použita pro vybrání *n* nejlepších tříd pro každý vektor. To je později použito pro predikci 10 nejlépe vyhovujících květin pro rozpoznání rostlin na webové databázi.

3.4 Struktura aplikace

V této části jsou popsány veškeré soubory aplikace pro rozpoznávání, včetně způsobu propojení s Node.js pomocí frameworku ZeroRPC. Dále jsou zde popsány metody načítání zdrojových obrazů a způsob uložení dat pro přenos mezi instancemi Python.

3.4.1 Modul supletools.py

Pro potřeby rozpoznávání byla vytvořena knihovna, která obsahuje užitečné metody pro segmentaci a práci s obrazem. Nachází se v souboru *supletools.py* a je používána ostatními programy pro načtení trénovacích nebo testovacích dat, uložení nebo načtení proměnných jazyka Python, maskování obrazu a další operace s obrazovými daty. Načtení obrazů z disku se dělí na dvě části – načítání z testovací databáze a z webové databáze.

Testovací databáze obsahuje obrazy 17 rostlin, 80 snímků každé rostliny a je

uložena v jedné složce souborového systému. Celkem je tedy pracováno s 1360 obrazy. Pro načtení je tedy třeba zvolit složku s obrazy a vygenerovat vektor tříd. Ten je generován pomocí cyklu *for* a označuje obrazy po 80 krocích vždy stejným číslem. Poté jsou všechny obrazy zkonvertovány na velikost 128*128 pixelů a jsou segmentovány metodou GrabCut. Algoritmus se dále dělí na dvě části – výpočet HOG a výpočet histogramu barev. Pro výpočet histogramu je obraz převeden do barevného prostoru HSV, poté jsou vybrány pouze pixely popředí a vektor je uložen do pole s příznaky barvy. Druhá část algoritmu převede obraz z prostoru RGB do stupňů šedi, a pokud byla segmentace úspěšná, výsledný obraz obsahuje více než 100 nenulových pixelů, je proveden výpočet HOG a vektor je vložen do pole s příznaky tvaru. Po skončení běhu metody jsou uživateli poskytnuty pole s vektory tvaru, barvy a vektor tříd.

Načtení obrazů webové databáze se liší pouze v úvodním načtení souborů, další operace jsou totožné jako při načítání testovací databáze. Rozdíl je jednak v umístění, ze kterého jsou zdrojová data načtena, ale i v typu o jaká data se jedná. Existují totiž dvě varianty – adresáře rostlin a stromů. Složky s názvy tříd se nacházejí ve stejné složce, a tak je bylo nutné odlišit. To je vyřešeno pomocí předpony „trees_“ před názvem stromu, rostliny mají název daný pouze úpravou latinského názvu. Poté jsou již načteny všechny obrazy k dané rostlině nebo stromu společně s uživatelem oříznutými fotografiemi ve složce „support_img“. Vektor tříd je vytvářen pomocí jednotlivých názvů složek, kdy obrazům z jedné složky je přiřazeno konkrétní indexové číslo a je vytvořen list názvů dle indexů rostlin.

Knihovna supletools obsahuje také metody pro uložení a načtení proměnných nebo celých objektů jazyka Python. To je využito hlavně při přenosu mezi procesy nebo pro zrychlení načítání dat. Metody pro načtení dat jsou pomalé a používají mnoho funkcí pro vytvoření příznakových vektorů. Proto je výhodné tyto vygenerované vektory uložit do souboru a pro testování optimálních parametrů klasifikátoru je načítat ze souboru. Tím je zvýšena rychlost výpočtu a jsou zaručeny stejné výsledky pro jednotlivé testy klasifikátorů. Další využití je pro použití na webovém serveru, kdy jeden skript natrénuje klasifikátory, PCA, klasterizační algoritmy a celé objekty uloží do souborů. Rozpoznávací aplikace pak již jen načte konkrétní objekty pro konkrétní typ rozpoznávání.

Další metody, které knihovna obsahuje jsou:

- **grabcut** – metoda pro výpočet segmentace algoritmem GrabCut, vstupem je dvourozměrný vektor obrazu v prostoru RGB. Výstupem je vysegmentovaný obraz, kde je nahrazeno pozadí černými pixely a ještě je uživateli vrácena maska, která označuje popředí a pozadí obrazu.
- **pca** – metoda PCA naprogramována podle popisu v kapitole 3.3.1. Vstupem je matice X, na které má být vypočítán výsledný prostor a počet komponent, které uživatel požaduje. Výstupem je pak vektor *eigenvalues*, který obsahuje vlastní čísla vektorů v matici eigenvectors. Ta je seřazena právě podle vlastních vektorů a je normalizována pomocí funkce *linalg.norm* z knihovny numpy. Posledním výstupním parametrem je průměrný vektor květiny.
- **clusterizeColors** – je metoda pro výpočet histogramu barev prostoru HSV redukovaného do 128 barev. Vstupem je pole objektů, kdy každý objekt reprezentuje jeden obraz a jeho vymaskované popředí. Proto je každý objekt jinak velký a je nutné ke každému objektu přistupovat zvlášť. Nejprve je potřeba natrénovat jádro algoritmu k-means. K tomu je použito 30 náhodně vybraných vektorů ze vstupního pole, které jsou spojeny za sebe do jednoho dlouhého vektoru a pomocí metody *fit* je provedeno trénování. Po natrénování je možné provést výpočet na každém vektoru vstupního pole a poté vypočítat histogram barev. Výsledkem je vektor se 128 hodnotami, které určují počty jednotlivých barev. Ten je ještě normalizován pomocí metody *linalg.norm* z knihovny numpy. Po dokončení klasterizace všech vektorů je objekt s klasterizátorem uložen do souboru, aby ho bylo možné použít při rozpoznávání.
- **maskImage** – převádí vstupní obraz na jednotlivé pixely dle masky tak, že pokud je maska rovna 1, pak pixely přidá do výstupního pole. Díky tomu je možné vybrat všechny pixely dle masky spočítané metodou GrabCut.
- **evaluateDecisionFunction** – slouží k vyhodnocení klasifikace metodou SVM. Analyzuje pole rozhodovacích funkcí a spočítá jednotlivé kladné hodnoty ke každé třídě klasifikátoru. Třída s nejvíce kladnými body je pak vyhodnocena jako nejlepší. Díky této metodě je možné sestavit chronologické seřazení tříd dle kladných bodů a tak vybrat například 10 nejlépe hodnocených tříd.

3.4.2 Trénování a klasifikace

Trénování může probíhat nad dvěma typy databází. Při použití testovací databáze jsou natrénovány klasifikátory a ihned otestovány náhodným výběrem vzorků z testovací databáze. V případě databáze webové jsou natrénovány klasifikátory, avšak tyto objekty jsou poté uloženy do souborů pro pozdější použití. V aplikaci jsou pro tyto účely vytvořeny dva soubory – *suple_svm.py* pro webovou databázi a *oxford_svm.py* pro testovací databázi. Testování probíhá zpravidla offline na stolním počítači a je zde optimalizováno nastavení parametrů, oproti webové databázi, kde má vše na starost serverový počítač. Oba soubory jsou tedy podobné struktury a obsahují následující postup trénování:

- načtení zdrojových obrazů a vytvoření vektorů příznaků,
- vytvoření histogramů barev pomocí metody clusterizeColors,
- výpočet PCA a převod příznakových vektorů do prostoru *eigenspace,*
- vytvoření 4 druhů příznakových vektorů (tvar rostliny, barva rostliny, tvar a barva rostliny, tvar stromu),
- vytvoření 4 různých SVM objektů a natrénování dle příslušných příznakových vektorů,
- uložení do souboru 4 objektů SVM, prostoru PCA pro rostliny a stromy, uložení objektu k-means klasterizace,
- v případě souboru testování je ještě vyhodnocena úspěšnost klasifikátoru.

3.4.3 Spouštěcí soubor aplikace

Pro spuštění aplikace pro rozpoznávání a pro komunikaci s webovým serverem byl vytvořen soubor python.py, který obsahuje metody používané webovou aplikací a kód pro vytvoření serveru zeroRPC. Nejprve je vytvořena třída s názvem HelloRPC, která obsahuje metody, které je možné volat vzdáleně. V této aplikaci je používána pouze jedna metoda – recognizeFile. Jako vstupní parametry jsou uvedeny název souboru a typ rozpoznávání. Poté je zahájeno rozpoznávání, které postupně načte obraz, zmenší ho a provede výpočet příznakových vektorů podle postupu popsaného v kapitole 3.2. Na základě zvoleného typu rozpoznávání je načten klasifikátor a klasterizátor a provedeno rozpoznávání. Výstupem metody je list 10 latinských názvů rostlin, které

byly rozpoznány s největší pravděpodobností.

Poslední částí souboru je vytvoření serveru zeroRPC pro vytvoření rozhraní pro vzdálené volání metod. Vytvoření objektu serveru je provedeno pomocí příkazu zerorpc.Server se specifikací třídy s metodami. Dále je nutné zvolit tcp adresu a port, v tomto případě je zvolena adresa 0.0.0.0, což znamená naslouchání na všech adresách a jako port je zvolen tcp 4242. Poté je již server spuštěn pomocí příkazu server.run.

4 Webový systém

V této kapitole byl navržen webový systém umožňující spojení s databází a s rozpoznávací aplikací pomocí knihovny zeroRPC. Dále jsou zde popsány všechny metody a soubory, které systém obsahuje a také ovládání systému.

4.1 Databáze MongoDB

Jako databáze byla zvolena aplikace MongoDB, která se hojně používá právě ve spojení s Node.js. Její instalace probíhá v režii operačního systému a poté je databáze připravena pro používání. Pro připojení k databázi je potřeba znát adresu a port služby, kde je databáze zpřístupněna a název databáze v aplikace MongoDB. Pro připojení k databázi je v systému použita knihovna Monk pro Node.js.

4.1.1 Kolekce rostlin

Pro uchovávání informací o rostlinách byla vytvořena kolekce dokumentů s názvem *flowers*, která obsahuje všechny rostliny. Informace ukládané ke každé rostlině jsou typu text a obsahují data o názvu rostliny, latinském názvu, popisu rostliny a adresáři se soubory. Takto navržená databáze obsahuje všechny důležité informace k rostlinám, které si uživatel může zobrazit po výsledném rozpoznávání.

4.1.2 Kolekce uživatelů

Pro potřeby zabezpečení aplikace a následnému ověřování uživatelů byla vytvořena kolekce *usercollection*. Ta obsahuje pouze minimální počet položek pro ověření uživatele – jméno a heslo.

4.2 Funkce webového systému

Webová aplikace byla vytvořena tak, aby nabízela uživateli pokročilé funkce webového systému, mezi které patří:

- zabezpečení portálu pomocí přihlašování uživatelů,
- přidávání rostlin do databáze a ukládání informací k nim,
- nahrávání obrazů,
- editace obrazů pomocí nástroje pro ořezávání,

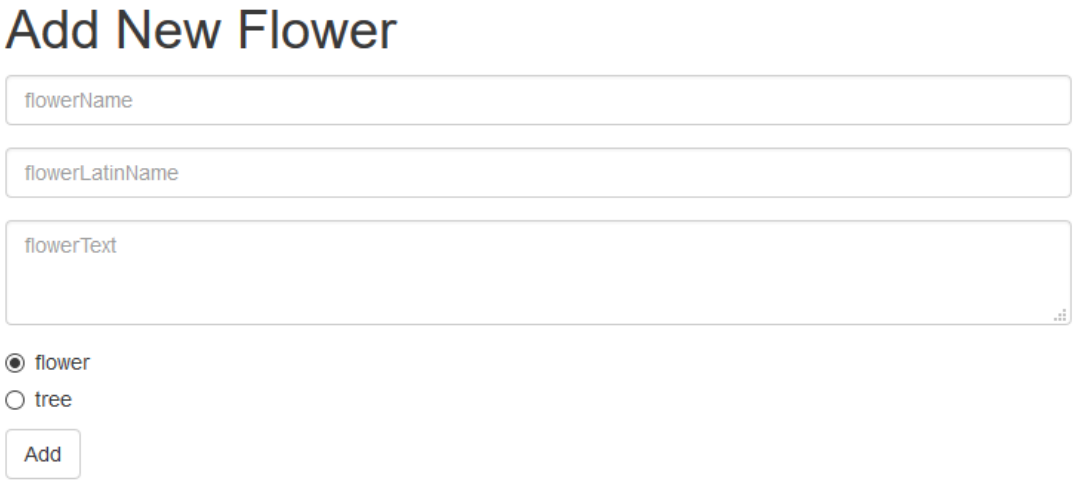
- možnost automatického rozpoznávání rostlin,
- automatické trénování klasifikátorů.

4.2.1 Správa rostlin

Pro pracování s rostlinami a jejich daty byla navržena struktura, která umožňuje ukládání dat o rostlinách do databáze pomocí webových stránek a ukládání obrazů k rostlinám přes webové rozhraní nebo pomocí protokolu pro přenos souborů. V následujících odstavcích jsou popsány metody pro práci s rostlinami.

Přidání rostliny do systému je realizováno pomocí formuláře, který obsahuje textová pole pro název rostliny, latinský název rostliny, popis rostliny a tlačítkový přepínač pro zařazení nové rostliny mezi databázi stromů nebo rostlin. Po odeslání formuláře je provedeno zpracování údajů. Nejprve je převeden textový řetězec latinského názvu tak, že jsou nahrazeny jiné znaky než písmena podtržítkem a poté je převedený textový řetězec použit jako název adresáře této rostliny. Poté je vytvořen záznam v databázi obsahující data z formuláře. Nakonec je vytvořen adresář rostliny. Pro použití této funkce byly naimplementovány dvě metody webového rozhraní:

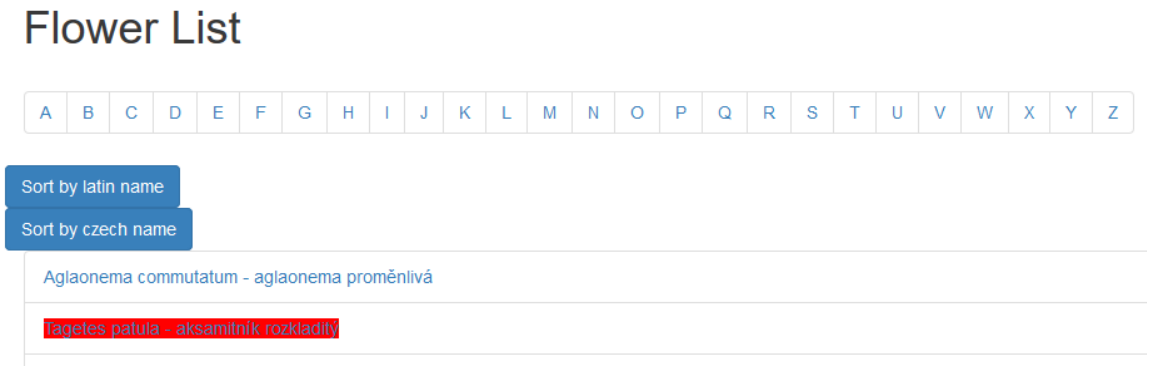
- GET /newflower – pro zobrazení formuláře,
- POST /addflower – pro zpracování dat z formuláře.



Obr. 20: Ukázka formuláře pro vložení nové rostliny

Po úspěšném přidání rostliny je tato rostlina zobrazena ve výpisu všech rostlin na adrese GET /flowers. Zde je možné výpis srovnat dle českého nebo latinského názvu, to je řešeno pomocí parametru sort. Dále jsou zde zvýrazněny rostliny, které ještě

nebyly uživatelem oříznuty, pro lepší přehled mezi již zpracovanými rostlinami. Dále je zde také implementována funkce pro zobrazení určité části rostlin, které začínají konkrétním písmenem.



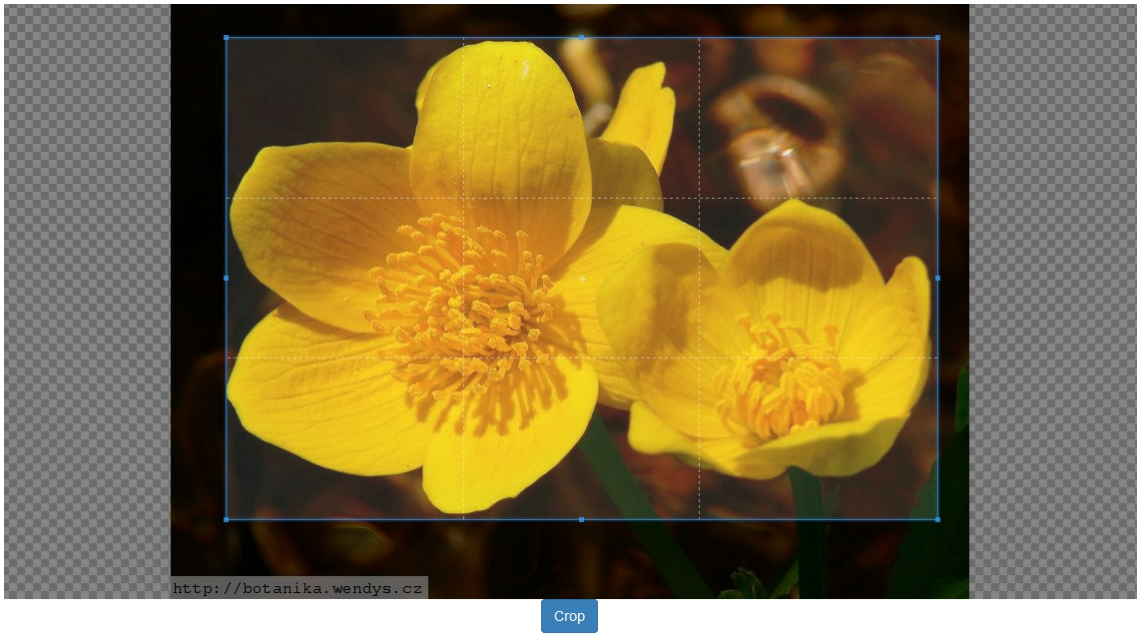
Obr. 21: Ukázka výpisu rostlin ve webové aplikaci

Pro zobrazení konkrétní rostliny stačí uživateli kliknout na odkaz z výpisu rostlin a je mu zobrazena stránka s konkrétní rostlinou. Adresa je určena předponou /flower a parametrem id, které má každá rostlina unikátní. Stránka s rostlinou obsahuje informace o rostlině, které jsou uloženy v databázi, dále výpis všech obrazů rostliny včetně upravených obrazů uživatelem. Tato stránka tedy slouží také jako galerie obrazů konkrétní rostliny, kam je možné nahrávat další fotografie. K tomu slouží formulář pro nahrání fotky, kde uživatel vybere fotografii ze svého počítače, a nahraje ji na server, a tam je uložena do adresáře rostliny. Nahrané fotografie je možné také mazat anebo je možné smazat i celou rostlinu.

Obr. 22: Galerie fotografií rostliny

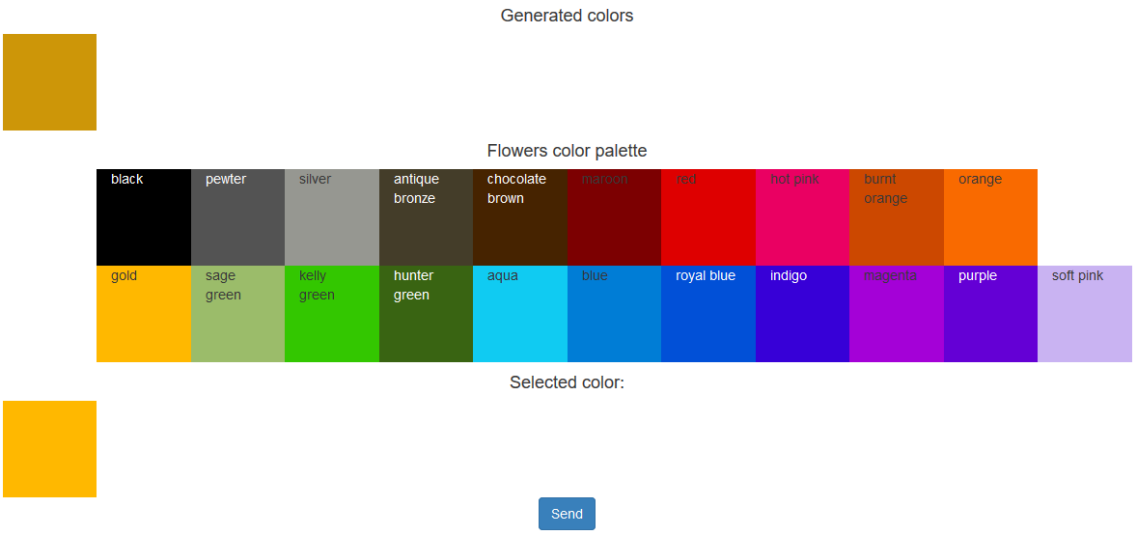
Dále je zde možnost vybrat jednu fotku z galerie, kterou lze následně oříznout pomocí nástroje *Cropper*. Označená fotografie z galerie je po kliknutí na tlačítko *select* zobrazena do *html* elementu *canvas* a je umožněno její oříznutí.

Na stránce pro oříznutí fotografie je použito knihovny Cropper, která je distribuována k uživateli ve formě javascriptového souboru. K aktivaci nástroje pro oříznutí je zapotřebí specifikovat *html* element, který má být tímto nástrojem používán. To je docíleno pomocí atributu *id* elementu *img*. Poté se již uživateli zobrazí element *canvas* s vykresleným obrazem a po najetí myší je možno oříznout obraz dle úvahy uživatele.



Obr. 23: Nástroj pro oříznutí rostliny

Po označení ořezávané plochy je po stisknutí tlačítka crop tato plocha vyříznuta a zobrazena na další stránce s výběrem barvy. Zde je nejprve spočítána nejvíce zastoupená barva v oříznutém obraze a k ní je dopočítána barva ze vzorové barevné palety. Uživatel může takto vygenerovanou barvu změnit tak, že klikne myší do obrazu na květ a je vypočítána nová barva tentokrát z okolí pozice kliku myši. Následně uživatel potvrdí výběr a odeslání tlačítkem *send*.

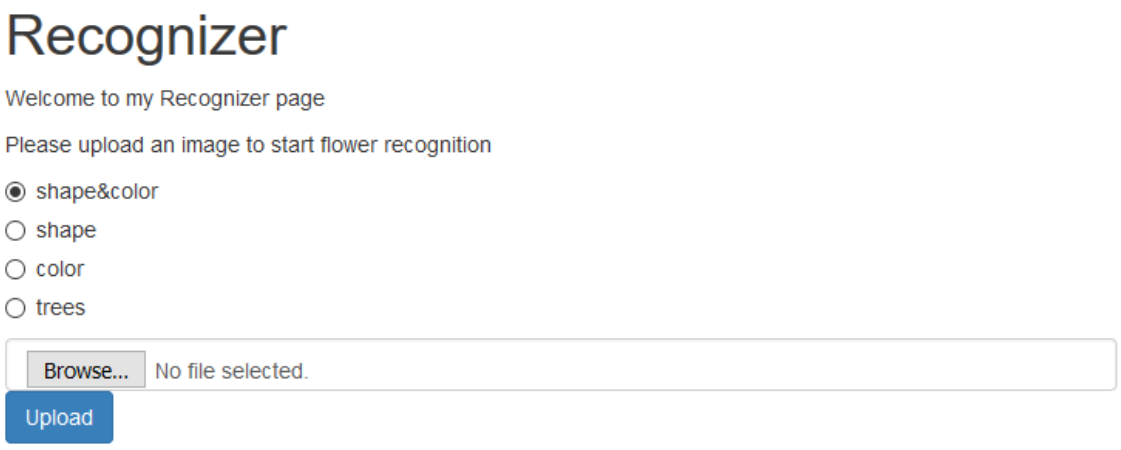


Obr. 24: Vygenerovaná barva a paleta barev

Data jsou na server odeslána v podobě objektu JSON a obsahují obraz ve formátu jpeg/base64, identifikátor rostliny, vybranou barvu uživatelem a zatřízenou barvu do barevné palety. Poté jsou barvy zpracovány a uloženy do databáze kromě obrazu, který je uložen do složky rostliny a podsložky pro uživatelsky oříznuté obrazy. Obraz je nutné neprve převést z formátu jpeg/base64 do jpeg, aby mohl být na serveru uložen.

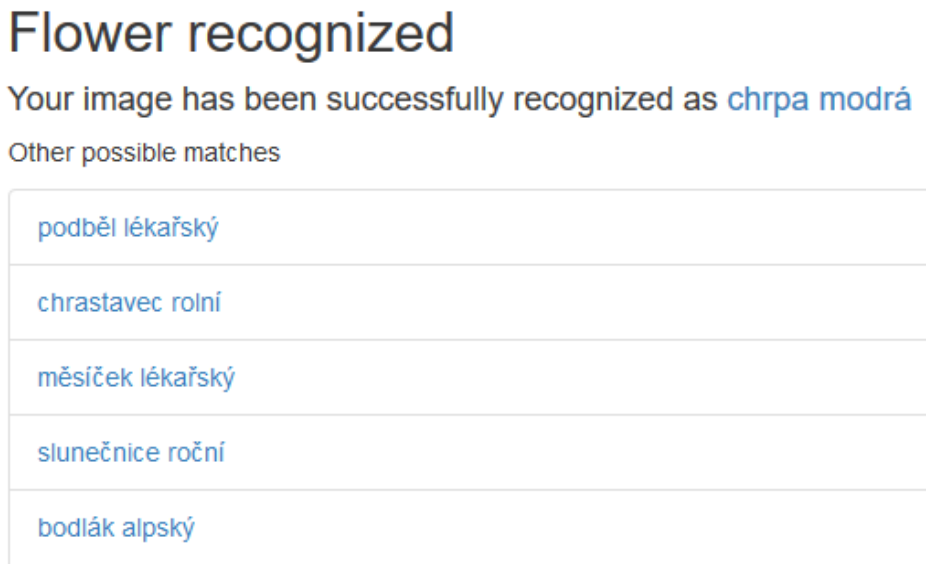
4.2.2 Rozpoznávání rostlin

Nejdůležitější funkcí webového systému je rozpoznávání obrazů nahraných uživatelem. K tomu slouží stránka na adrese GET /recognize, která uživateli zobrazí formulář pro nahrání obrazu, a možnost vybrat ze 4 typů rozpoznávání viz obr. 23.



Obr. 25: Formulář pro nahrání souboru a rozpoznání

Po odeslání formuláře je soubor přijat a uložen na serveru do složky pro obrazy připravené pro rozpoznávání. Následně je odeslán požadavek aplikaci pro rozpoznávání pomocí knihovny zeroRPC a příkazu invoke. Po obdržení odpovědi s výsledky rozpoznávání jsou tato data použita pro dotaz do databáze a postupně jsou načtena data ke všem deseti výsledkům rozpoznávání. Poté jsou výsledky zobrazeny uživateli a ten si může jednotlivé rostliny vybrat a zobrazit jejich informace z databáze.



Obr. 26: Ukázka výsledku rozpoznávání

4.2.3 Zabezpečení systému

Webová aplikace je zabezpečena proti neověřeným uživatelům pomocí přihlašování. Přihlášení je možné uskutečnit na hlavní stránce nebo na adrese /login. Formulář pro přihlášení má pouze dvě textové pole – jméno a heslo, dále obsahuje tlačítko pro odeslání údajů. Ty jsou po odeslání na server ověřeny pomocí knihovny Passport.js dotazem do databáze a porovnáním obou textových polí. Pokud je nalezena shoda ve jménu a hesle, je uživatel přihlášen a vygenerován speciální řetězec, který je uživateli vložen do *cookies* a slouží jako *session* pro ověření trvalého přihlášení. To znamená, že se uživatel přihlásí pomocí formuláře a poté je přihlášen na celé aplikaci.

Obr. 27: Přihlášení uživatele

5 Testování algoritmu

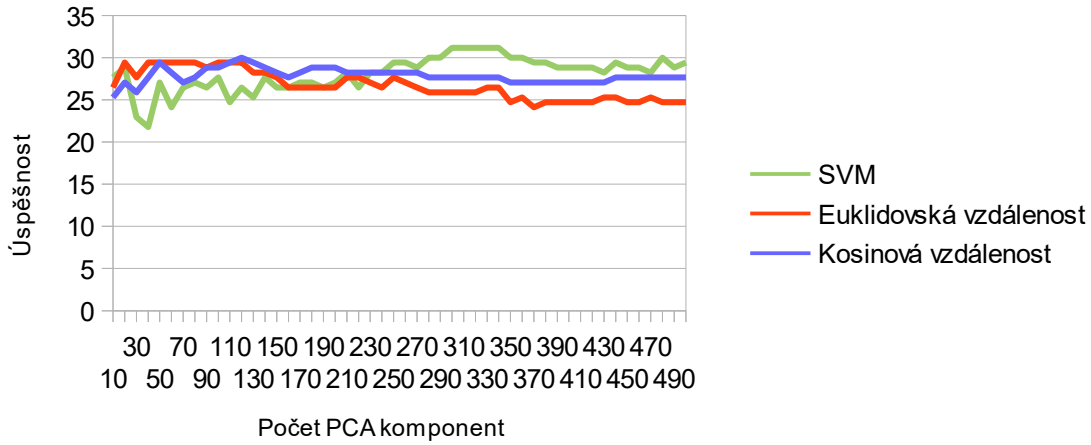
V této kapitole je popsán způsob testování úspěšnosti metod použitých pro rozpoznávání rostlin. Jako databáze byla použita Oxfordská databáze 17 rostlin, která je popsána v kapitole 2.2.2. Nejprve byla testována metoda PCA použitá s různými klasifikačními algoritmy na nesegmentovaných datech. Poté byla vyzkoušena metoda segmentace a její různá nastavení. Další krok byl přidání metody pro popis tvaru – LARK a HOG. V závěru je pak srovnání jednotlivých metod popisu tvaru a barvy.

5.1 Metoda PCA

Pro testování metody PCA byla načtena data z testovací databáze a převedena do šedotónového barevného prostoru. Jedná se tedy o nevysegmentované obrazy rostlin, které obsahují pozadí i popředí. Po převedení vektorů obrazů do prostoru eigenspace pomocí metody PCA bylo nutné vyzkoušet klasifikační algoritmy. Ty byly zvoleny následující:

- Klasifikace vzdálenostní funkcí – eukleidovská a kosinová
- Klasifikace algoritmem SVM

Použitím výše zmíněných metod bylo dosaženo nejvyšší úspěšnosti pomocí SVM, a to **31,2 %**. Srovnání jednotlivých metod je znázorněno na Obr. 26.



Obr. 28: Srovnání metod klasifikace PCA

Z obrazu je možné vidět, že jednodušší metody výpočtu vzdálenosti jako jsou eukleidovská a kosinová vzdálenost při malém počtu komponent mají vyšší výsledky než SVM. Ta naopak s přibývajícími PCA komponenty svou úspěšnost zvyšuje a při 300 komponentách má úspěšnost nejvyšší. Parametr C byl určený experimentálně a jako nejlepší byla určena hodnota C=0,0000001. Proto byla vybrána jako hlavní klasifikátor pro tuto práci, společně s určujícím parametrem 300 hlavních komponent.

5.2 Segmentace GrabCut

Pro segmentaci obrazu byl použit algoritmus Grabcut, který má dva parametry nastavení a ty bylo nutné optimalizovat. Jedním z nich je obdélník pro určení pozadí a popředí rostliny. Ten byl určován tak, že byl nejprve stejně velký jako vstupní obraz a poté o jeden pixel zmenšován. Tím dochází k větší segmentaci a tudíž ke zpřesňování ořízntutého květu. Další parametr je počet iterací algoritmu, který stejně jako zmenšování obdélníku pro popis popředí, určuje míru segmentace, kdy více iterací znamená větší segmentaci. Srovnání různých nastavení je v tabulce 1.

PCA+GC	Iterace 1	Iterace 2	Iterace 3
1 pixel	25,88%	28,99%	29,58%
2 pixely	26,54%	**32,28%**	31,85%
3 pixely	27,22%	29,30%	29,30%

Tabulka 1: Parametry metody GrabCut

Z tabulky je možné vidět, že nejlepších výsledků bylo dosaženo s obdélníkem o 2 pixely menším než je původní obraz a 2 iteracích algoritmu. Byly testovány i další kombinace parametrů, v tabulce jsou uvedeny jen ty nejdůležitější a nejúspěšnější.

Klasifikátor SVM byl spouštěn s parametrem C=0,0000001 a 300 hlavními komponentami. Oba parametry byly určeny experimentálně pro maximální úspěšnost.

5.3 Rozpoznávání podle tvaru

Pro vytvoření příznakových vektorů z obrazu byly vyzkoušeny 2 algoritmy. Nejdříve byl vyzkoušen LARK, který je schopen klasifikovat rostliny s úspěšností přes 40 %. Jako druhý byl aplikován na segmentovaný obraz HOG, který byl poté na základě výsledků úspěšnosti vyhodnocen jako nejlepší s úspěšností přesahující 55 %.

5.3.1 LARK

Nejprve byla vyzkoušena metoda LARK z článku [7], kde je uvedena jako nejúspěšnější metoda pro verifikaci osob za použití obličejů. V této aplikaci nebyla nejúspěšnější, avšak má vyšší úspěšnost rozpoznávání než pouze vysegmentovaná rostlina. Jako parametry pro metodu LARK byly použity ukázkové hodnoty z článku. Úspěšnosti této metody v závislosti na úrovni segmentace popisuje tabulka 2.

	Iterace 1	Iterace 2	Iterace 3	Iterace 4	Iterace 5
1 pixel	42,30%	**42,60%**	40,00%	41,40%	42,00%
2 pixely	40,00%	36,90%	39,40%	37,90%	35,40%

Tabulka 2: Úspěšnost LARK v závislosti na úrovni segmentace GrabCut

Parametr C klasifikátoru SVM byl zvolen experimentálně a to 1,0 a počet PCA komponent 300.

5.3.2 HOG

Jako druhá metoda pro popis tvaru květiny byla zvolena metoda HOG. Používá pro vytvoření příznakových vektorů histogramy orientovaných gradientů. Tato metoda byla otestována na různé úrovně segmentace, nastavení parametrů klasifikátoru a různý počet PCA komponent.

Počet PCA komponent vyšel nejlépe 300, vzhledem k nižším hodnotám úspěšnosti u malého počtu komponent a k velice podobným hodnotám úspěšnosti u počtu komponent nad 100.

Parametr klasifikátoru SVM – C byl volen na základě úspěšnosti a testován vždy od hodnoty 1000000 až 0.00000001 s krokem 0.1. Jako nejlepší se jevily parametry 0.1 a 0.01.

Úspěšnost se lišila hlavně v závislosti na úrovni segmentace, jak ukazuje tabulka 3.

	Iterace 1	Iterace 2	Iterace 3	Iterace 4	Iterace 5	Iterace 6
1 pixel	52,35%	53,25%	54,44%	50,89%	52,66%	50,89%
2 pixely	52,17%	**55,97%**	54,72%	55,06%	55,06%	54,43%
3 pixely	52,83%	54,72%	53,85%	54,84%	54,72%	54,84%

Tabulka 3: Úspěšnost HOG v závislosti na úrovni segmentace GrabCut

Z tabulky je možné vidět, že nejlepší hodnotu 55,97 % má tento algoritmus při použití segmentovaného obrazu metodou GrabCut s nastavením obdélníků o velikosti 2 pixely menší než obraz a 2 iteracemi. Pro kontrolu byly všechny takto vysegmentované obrazy z testovací databáze uloženy a vizuálně zkontrolovány. Dle hodnocení přesnosti segmentace bylo zkontrolováno, že toto nastavení je z hlediska segmentace nejlepší.

5.4 Rozpoznávání podle barvy

Pro popis barev květu byl vybrán histogram z redukovaného prostoru barev na 128. Redukci provádí algoritmus k-means v barevném prostoru HSV. Nejprve je rostlina segmentována na květ a pozadí, jsou extrahovány pixely květu a proveden výpočet histogramu. Je tedy možné volit pouze parametr určující počet košů histogramu a počet klasterů algoritmu k-means. Obě hodnoty musí být stejné. Srovnání úspěšností popisuje tabulka 4.

Poč. košů	8	16	32	64	128	256	512
úspěšnost	40,88%	41,51%	44,02%	43,40%	**49,06%**	49,06%	49,06%

Tabulka 4: Úspěšnost klasifikace rostlin podle barvy

Z tabulky je patrné, že s postupně přibývajícími počty košů se zvyšuje úspěšnost. Nejvyšší úspěšnosti bylo dosaženo při počtu 128 košů a s více koši již úspěšnost nestoupá. Pro popis barvy rostliny bylo tedy zvoleno nastavení 128 barev.

5.5 Rozpoznávání podle barvy a tvaru

Poslední testovaná varianta příznakového vektoru spočívá ve spojení vektoru tvaru a barvy. Jednotlivé vektory byly spojeny do jednoho a poté použity pro trénování klasifikátoru. Výsledky klasifikace jsou uvedeny v tabulce 5.

Algoritmus	Úspěšnost
PCA	31,20%
PCA+GC	32,28%
PCA+GC+LARK	42,60%
PCA+GC+HSV	49,06%
PCA+GC+HOG	55,97%
PCA+GC+HOG+HSV	57,59%

Tabulka 5: Celková úspěšnost systému pro jednotlivé metody

Závěr

V rámci této diplomové práce byly vybrány a otestovány algoritmy pro automatické rozpoznávání rostlin v obraze. Byly porovnány metody pro segmentované a nesegmentované obrazy, stejně jako různé metody pro popis příznaků objektů v obraze. Nejprve byla použita metoda PCA s obrazy z testovací databáze bez segmentace. Pro klasifikaci byly postupně vyzkoušeny 3 různé klasifikace – eukleidovská vzdálenost, kosinová vzdálenost a SVM. Nejvyšší úspěšnost dosáhla metoda SVM **31,2 %** a byla zvolena pro klasifikaci i ostatních obrazů. Pro zpřesnění obrazu květu rostliny byly vyzkoušeny 2 metody segmentace a vybrána metoda GrabCut na základě vizuálního posouzení. V kombinaci s redukcí dimenze PCA a klasifikací pomocí SVM dosáhla úspěšnosti **32,28 %**. Dalším krokem bylo vyzkoušení algoritmů pro popis tvaru květu v obraze – LARK a HOG. První algoritmus pro různá nastavení segmentace dosáhl nejvyšší úspěšnosti **42,60 %**. Druhý algoritmus dokázal klasifikovat data s úspěšností **55,97 %** a byl vyhodnocen jako nejlepší. Pro popis barvy rostliny v obraze byl spočítán histogram obsahující 128 košů a hodnoty barev z barevného prostoru HSV. Pro redukci počtu barev byl použit algoritmus k-means. Výsledná úspěšnost klasifikace rostlin na základě barvy dosáhla **49,06 %**. Pro konečné rozpoznávání rostlin v obraze byly příznaky tvaru a barvy spojeny do jednoho vektoru a použity pro klasifikaci s úspěšností **57,59 %**.

Takto navržený a otestovaný algoritmus byl propojen s webovou aplikací umožňující online rozpoznávání rostlin v obraze. Rozpoznání probíhá na základě obrazu nahraného uživatelem do systému a je zobrazeno 10 nejlépe klasifikovaných rostlin. Webový systém umožňuje uživateli vytvářet nové rostliny a nahrávat k nim nové obrazy. Tyto obrazy lze zpřesnit pomocí nástroje pro oříznutí a tím vybrat pouze oblast s květem nebo listem. Následně je možné přetrénovat klasifikační algoritmy s pomocí databáze obsahující nově vytvořené obrazy.

Seznam použité literatury

[1] BRADSKI, G. a A. KAEHLER. Learning openCV: computer vision in C with the OpenCV library. 1. Sebastopol: O'Reilly Media, Inc, 2008. ISBN 978-1-4493-1465-1.

[2] ŠONKA, M., V. HLAVÁČ a R. BOYLE. Image processing analysis and machine vision. London: Chapman, 1993. Chapman. ISBN 04-124-5570-6.

[3] NOCK, R. a F. NIELSEN. Statistical region merging. IEEE Transactions on Pattern Analysis and Machine Intelligence [online]. 2004, 26(11), 1452-1458 [cit. 2016-05-08]. DOI: 10.1109/TPAMI.2004.110. ISSN 0162-8828. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1335450

[4] ROTHER, C., V. KOLMOGOROV a A. BLAKE. "GrabCut". ACM SIGGRAPH 2004 Papers on - SIGGRAPH '04 [online]. New York, New York, USA: ACM Press, 2004, , 309- [cit. 2016-05-08]. DOI: 10.1145/1186562.1015720. Dostupné z: http://portal.acm.org/citation.cfm?doid=1186562.1015720

[5] SURAL, S., GANG QIAN a S. PRAMANIK. Segmentation and histogram generation using the HSV color space for image retrieval. Proceedings. International Conference on Image Processing [online]. IEEE, 2002, , II-589-II-592 [cit. 2016-05-14]. DOI: 10.1109/ICIP.2002.1040019. ISBN 0-7803-7622-6. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1040019

[6] TURK, M.A. a A.P. PENTLAND. Face recognition using eigenfaces. Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition [online]. IEEE Comput. Sco. Press, 1991, , 586-591 [cit. 2016-05-08]. DOI: 10.1109/CVPR.1991.139758. ISBN 0-8186-2148-6. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=139758

[7] SEO, H. J. a P. MILANFAR. Face Verification Using the LARK Representation. IEEE Transactions on Information Forensics and Security [online]. 2011, 6(4), 1275-1286 [cit. 2016-05-03]. DOI: 10.1109/TIFS.2011.2159205. ISSN 1556-6013. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5872024

[8] DALAL, N. a B. TRIGGS. Histograms of Oriented Gradients for Human Detection. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) [online]. IEEE, 2005, , 886-893 [cit. 2016-05-08]. DOI: 10.1109/CVPR.2005.177. ISBN 0-7695-2372-2. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1467360

[9] WEI-PING, Z., L. MING-XIN a C. HUAN. Using MongoDB to implement textbook management system instead of MySQL. 2011 IEEE 3rd International Conference on Communication Software and Networks [online]. IEEE, 2011, , 303-305 [cit. 2016-04-19]. DOI: 10.1109/ICCSN.2011.6013720. ISBN 978-1-61284-485-5.

Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6013720

[10] Herbář Wendys [online]. 2016 [cit. 2016-05-15]. Dostupné z: http://botanika.wendys.cz/

[11] 17 Category Flower Dataset. Visual Geometry Group, Department of Engineering Science, University of Oxford [online]. 2009 [cit. 2016-05-15]. Dostupné z: http://www.robots.ox.ac.uk/~vgg/data/flowers/17/

[12] NILSBACK, M-E. a A. ZISSERMAN. Automated Flower Classification over a Large Number of Classes. 2008 Sixth Indian Conference on Computer Vision, Graphics [online]. IEEE, 2008, , 722-729 [cit. 2016-05-15]. DOI: 10.1109/ICVGIP.2008.47. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4756141

Obsah přiloženého DVD

- Text diplomové práce ve formátu PDF
- Soubory aplikace pro rozpoznávání psané v jazyce Python
- Soubory webové aplikace psané v jazyce JavaScript