ČVUT v Praze

Listy – Podobnost jednoduchých tvarů převedených na časové řady

Matěj Kříž, krizmate

27. ledna 2016

Listy – aplikace pro rozpoznávání stromu podle tvaru listu

1. Popis projektu

Cílem projektu je vytvořit aplikaci demonstrující použití metody DTW pro měření podobnosti jednoduchých tvarů převedených na časové řady. Aplikace je určená primárně pro mobilní zařízení, ale je přístupná i v prohlížeči. Umožňuje načíst snímek listu, extrahovat obrys listu, převést ho na časovou řadu a tu pak porovnávat se záznamy v databázi. Nejpodobnější záznam aplikace ukáže uživateli.

2. Způsob řešení

Detekce hran

Pro detekci hran byl použit Cannyho hranový detektor („canny“). Před použitím canny je fotografie převedená do stupňů šedi a šum je eliminován Gaussovým filtrem (parametrizovatelným z rozhraní aplikace).

Detekce kontur

Protože zvolená knihovna pro detekci hran vrací výsledek v podobě matice se všemi nalezenými hranami, je ještě třeba z těchto hran vybrat obrys listu.

Hledání obrysu listu se provádí procházením matice s hranami. Jakmile je detekováno, že navštívený bod náleží nějaké hraně, algoritmus nejprve zkontroluje, zda tento bod ještě není přiřazen nějaké hraně. Pokud ne, tak tuto hranu sleduje hledáním sousedních pixelů v okolí. Z takto nalezených kontur bereme za obrys listu tu nejdelší, což funguje dobře, pokud list zaujímá většinu pořízené fotografie. Algoritmus je možné zjednodušeně popsat v pseudokódu takto:

```
findContours(imageMatrix, foregroundColor, backgroundColor,
threshold) {
for (y: array[0..height]) {
     for (x: array[0..width]) {
          if value > threshold then value = 255 else value = 0
```

```
            if (
                prevValue === backgroundColor
                && value === foregroundColor
                && pixel is not in any contour yet
            ) {
                points = followContour from point x, y
                add points to allContours
            }
        }
}
}

followContour(startPoint) {
    points[0] = startPoint
    while point != startPoint && pointsCount < maxContourPoints {
        newPoint = find next point in neighborhood
        if newPoint is not found then break
    }
    close contour
}
```

DTW

Dynamic time warping (česky metoda dynamického borcení časové osy) je algoritmus umožňující porovnávat dvě sekvence, ve kterých se mohou vzory lišit v časovém posunutí či rychlosti. Vlastní implementace je přesně pospána následujícím pseudokódem:

```
DTWDistance(s: array [1..n], t: array [1..m], w: int) {
    DTW := array [0..n, 0..m]
   for i := 0 to n
       for j:= 0 to m
           DTW[i, j] := infinity
    DTW[0, 0] := 0
```

```
    for i := 1 to n
        for j := max(1, i-w) to min(m, i+w)
            cost := d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j  ],
                                        DTW[i, j-1],
                                        DTW[i-1, j-1])
    return DTW[n, m]
}
```

3. Implementace

Celá aplikace je psána v jazyce JavaScript, tedy jak klientská část (frontend), tak serverová část (backend).

Frontend

Pro frontend je použit framework Ionic, jakožto nadstavba frameworku Angular poskytující připravené komponenty pro vývoj mobilních aplikací.

Detekce hran v obrazu probíhá na frontendu, aby fungovala interaktivně bez nutnosti přenášet pořízenou fotografii na server a bylo tak možné měnit vstupní parametry pro dosažení lepšího výsledku. Z toho důvodu nebylo možné použít knihovnu OpenCV, která implementuje velmi výkoný canny detektor vracející rovnou jednotlivé kontury v jazyce C. Proto byla nakonec použita knihovna jsfeat.

Pro detekci jednotlivých kontur byla vzata implementace z https://github.com/Doodle3D/Contour-finding-experiment, kterou bylo třeba optimalizovat pro hledání kontur listu.

Backend

Pro implementaci serveru byl použit framework Express, umožňující rychlé vytvoření REST API.

Původně byla na serveru zpřístupněna knihovna OpenCV, ale ve výsledné aplikaci je odstraněna, neboť funkce detekce hran byla přesunuta na frontend.

Algoritmus DTW je implementován vlastní, ale pro srovnání byla použita knihovna https://github.com/langholz/dtw.git.

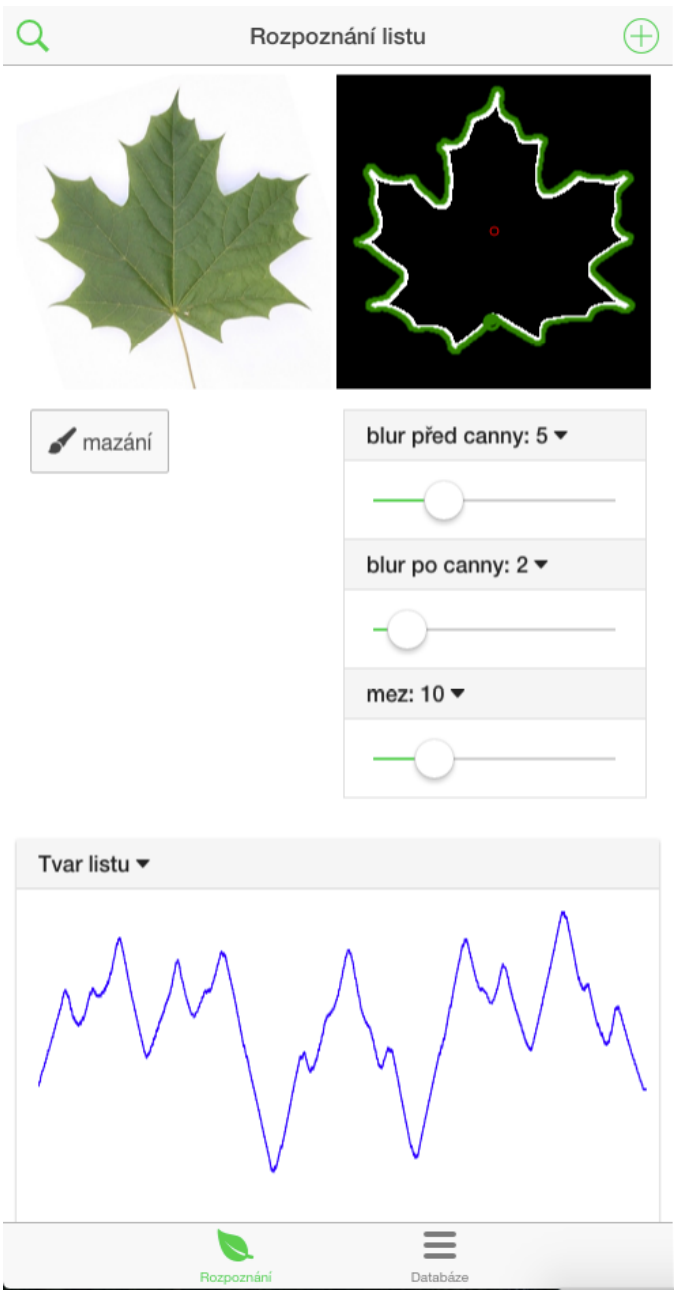
Pro uložení katalogu stromů byla použita databáze MongoDB.

Pro běh aplikace je potřeba mít nainstalováno:

- Node.js (v5.2.0)
- MongoDB (v3.2)
- gulp.js
- nodemon

Instalace a spuštění aplikace je popsána v souboru README.md v repozitáři aplikace.

Příklad výstupu



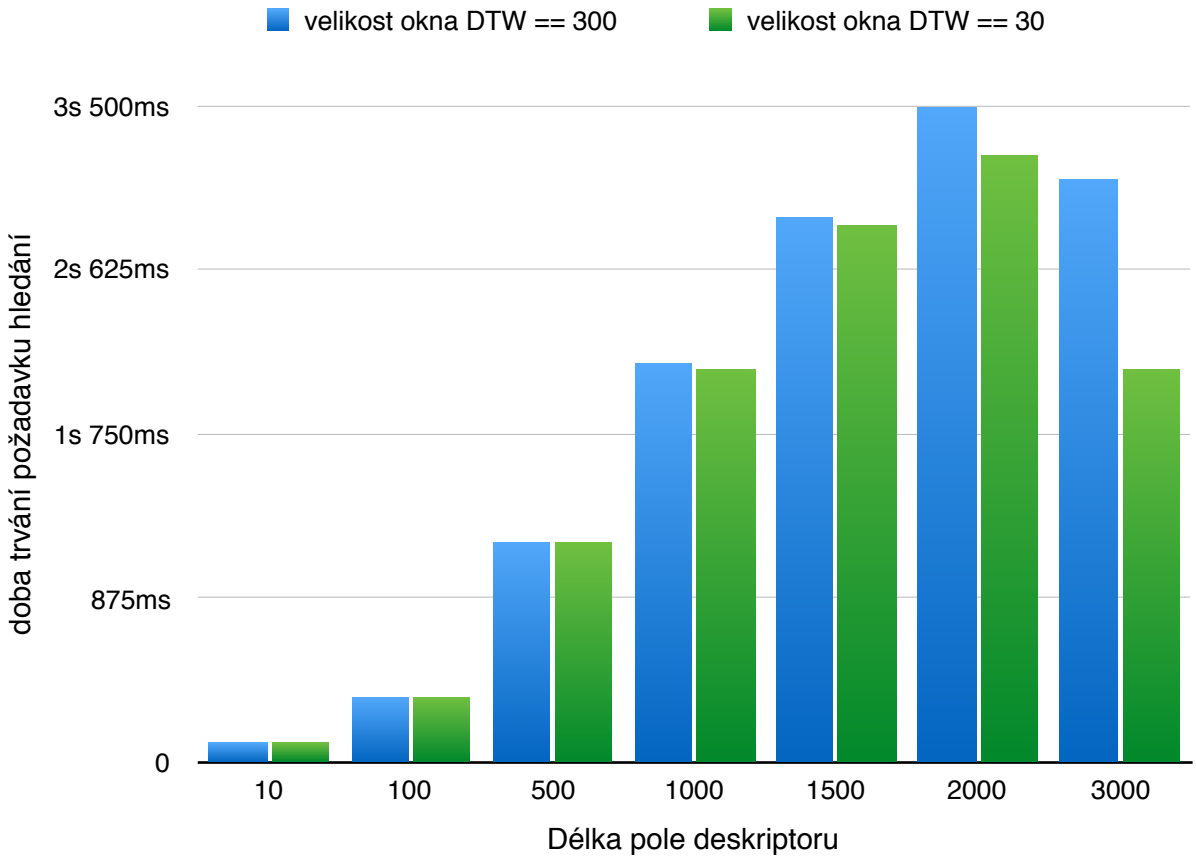
Obr. č. 1 a 2: Na obrázku vlevo (č. 1) je vidět stav aplikace po načtení fotografie listu, obrázek vpravo (č. 2) ukazuje stav aplikace po úspěšném dokončení hledání tvaru listu z obrázku č. 1.

```
api:trees treename: DTWDistance +13s Bříza :  251.46503608545282
api:trees treename: DTWDistance +686ms Buk lesní :  287.7448936223795
api:trees treename: DTWDistance +357ms Javor babyka :  176.10552170957757
api:trees treename: DTWDistance +377ms Javor Klen :  106.12635303549023
api:trees treename: DTWDistance +303ms Javor mléč :  31.004001193132552
api:trees treename: DTWDistance +323ms Dub letní :  455.55129530289247
api:trees treename: DTWDistance +295ms Lípa malolistá (srdčitá) :  224.86847707904798
api:trees treename: DTWDistance +339ms Habr obecný :  325.011651732519
api:trees minimal DTWDistance:  +0ms 31.004001193132552
```

Obr. č. 3: Konzolový výstup serveru v debug módu při hledání z obrázku č. 1.

Experimentální sekce

Proces výpočtu podobnosti v aplikaci má několik parametrů. Prvním zásadním pro přesnost hledání a dobu trvání dotazu je délka deskriptoru popisujícího funkci tvaru listu. Dalším, který se ukázal jako méně zásadní pro dobu trvání dotazu je velikost okna pro DTW. Na následujícím grafu jsou výsledky měření s různými hodnotami právě těchto parametrů. Výsledné doby trvání dotazů jsou průměrem z pěti měření na fotografiích třech různých listů.



Jistě by bylo zajímavé měření z hlediska přesnosti výsledků, ale nepodařilo se vymyslet takový experiment, který by měl vypovídající hodnotu. Výsledek vždy nejvíce záleží na vkládané fotografii a už od délky deskriptoru 100 se výsledky pro danou fotografii shodují (ať už jsou správné či nikoliv).

4. Diskuze

Největší prostor pro zlepšení kvality vyhledávání je při detekci hran. Tu nyní není možné provést, pokud list není vyfocen na jednolitém pozadí. Už teď je v aplikaci možná základní editace (otočení, zvětšení, posunutí, mazání), ale ovládání těchto funkcí by se mělo vylepšit.

Velké rozdíly způsobuje řapík, který je proto nutné z fotek umazávat. To je v současné verzi možné jen na fotografiích s bílím pozadím. Odstranění řapíku by ale bylo možné provádět automaticky [Ivana Váňová, Diplomová práce: Rozpoznávání druhů dřevin rostoucích v České republice podle tvaru listu, 2013, https://dip.felk.cvut.cz/browse/pdfcache/vanoviva_2013dipl.pdf].

Backend by bylo především potřeba optimalizovat z hlediska rychlosti výpočtu DTW.

Dále by měla být doplněna databáze stromů a u existujících záznamů doplnit kontury více listů, neboť některé listy i na jednom stromu se mohou zásadně lišit.

Pro reálnou aplikaci by také bylo vhodné doplnit více druhů deskriptorů a nespoléhat se pouze na tvar listu. Například velikost listu a barvu. Případně přidat detekci pomocí klasifikátoru neuronové sítě.

5. Závěr

Aplikace je schopna rozpoznat botanický druh stromu podle fotografie jeho listu, avšak pouze v ideálních podmínkách a zatím pouze u velmi malého počtu druhů.