VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PREDIKCE VÝVOJE SRÁŽEK Z METEOROLOGICKÉHO RADARU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MATEJ TÁBI
AUTHOR

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PREDIKCE VÝVOJE SRÁŽEK Z METEOROLOGICKÉHO RADARU

PRECIPITATION FORECAST FROM WEATHER RADAR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MATEJ TÁBI
AUTHOR

VEDOUCÍ PRÁCE Ing. RADEK BURGET, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Tato práce řeší webovou aplikaci s účelem krátkodobé předpovědi srážek pro území České republiky. Předpověď je vytvořena na jednu hodinu dopředu a má podobu čtyř radarových snímků, mezi kterými je 15 minutový odstup.
Zvolený problém je řešen ve třech krocích. V prvním kroku se získávájí aktuální údaje z webu ČHMÚ. Ve druhém kroku jsou zpracovávána data - identifikovaná srážková pole a ve třetím kroku jsou vypočteny dráhy pohybu jednotlivých polí, na základě nichž je vytvořena výsledná předpověď.
Výsledkem této práce je plně funkční webová aplikace. Přesnost předpovědi se pohybuje na úrovni od 30 do 60%.

Abstract

The aim of this bachelor thesis is a web application, which solves a short-time precipitation forecast for Czech Republic. The forecast is created for the next one hour and consists of four radar frames with a 15 minutes distance between them.
The task is solved in three main steps. In the first step, the actual information from CHMI webpages is obtained. In the second step, the recieved data is processed - precipitation fields are identified and in the last step, movement vectors of these precipitation fields are calculated.
The result of the thesis is a fully working web application. The forecast success rate is at the level from 30 to 60%.

Klíčová slova

počasí, předpověď počasí, node.js, webová aplikace, srážky, shlukovací metody

Keywords

weather, weather forecast, node.js, web application, precipitation, clustering methods

Citace

Matej Tábi: Predikce vývoje srážek z meteorologického radaru, bakalářská práce, Brno, FIT VUT v Brně, 2014

Predikce vývoje srážek z meteorologického radaru

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana inženiera Radeka Burgeta. Další informace mi poskytlol pracovník ČHMÚ, doktor Petr Novák. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Matej Tábi
20. května 2014

Obsah

Kapitola 1

Úvod

Numerická predpoveď počasia je proces, ktorý spracováva a prepočítava fyzikálne javy v atmosfére. Výpočetne je tento proces náročný a výsledky stále nie sú stopercenté, čo dáva priestor na zlepšovanie. Predpoveď zrážok patrí medzi hlavné súčasti predpovedí, snáď každého zaujíma predtým ako vyjde z domu či si má zobrať so sebou aj dáždnik a pršiplášť, alebo mu stačia ponožky v sandáloch.

Aplikácia, ktorá je výsledkom bakalárskej práce pojednáva o tejto problematike z krátkodobého hľadiska, vytvára predpoveď na nasledujúcu 1 hodinu. Treba ale zdôrazniť, že klasické predpovedné modely zrážok, napr. model Aladin, využívajú rôzne namerané hodnoty pri zostavovaní výstupu (teplotu, vlhkosť, tlak, rýchlosť verta, smer vetra a mnoho ďalších). Táto aplikácia preto nefunguje ako klasický numerický predpovedný model. Zbiera a zisťuje dáta iba analýzou najnovších radarových snímkov, ktoré sú k dispozícii na webovom portáli Českého hydrometeorologického ústavu. Preto sa jedná v podstate o získavanie znalostí z obrázkov a následné predpovedanie možného vývoja zo získaných dát.

Zrážky možu byť rôzneho charakteru ako napríklad oklúzny front (velmi pomalý postup a veľká plošná rozloha), teplý/studený front (rýchly postup zrážok), letné búrky (takmer bez pohybu). Veľmi dobré výsledky boli dosiahnuté pri stredne rýchlo postupujúcich zrážkach na studených alebo teplých frontoch, kde sa charakter zrážkových polí mení minimálne a viac-menej sa mení iba ich poloha.

Vývoj tejto aplikácie pozostáva z niekoľkých hlavných krokov, ktoré budú podrobnejšie popísané v nasledujúcich kapitolách. Budú popísané meteorologické prvky súvisiace s danou témou, implementačné prostredie a použité algoritmy, ktoré vytvárajú ucelený výpočet krátkodobej predpovede. Na záver budú spomenuté úskalia, ktoré boli nutné riešiť a samozrejme plne funkčná webová aplikácia.

Kapitola 2

Teoretické hľadisko

2.1 Zaradenie z meteorologického hľadiska

Numerická predpoveď počasia je definovaná ako predpoveď polí meteorologických prvkov, ktorá je výsledkom časovej integrácie prognostických rovníc niektorého fyzikálneho modelu atmosféry, vykonávanej na samočinných počítačoch metódami numerickej matematiky.[18]

Základom modernej predpovede počasia je práve numerická predpoveď (NWP). Schopnosť NWP simulovať fyzikálne a dynamické podmienky atmosféry sa neustále zvyšuje. Vznik a vývoj nebezpečných poveternostných javov je však často spôsobený lokálnymi vplyvmi. Procesy takejto priestorovej a časovej mierky sú v súčasných modeloch parametrizované a vlastný dej teda nie je modelovaný. Navyše fyzikálna parametrizácia popisuje proces iba približne. Krátkotrvajúce javy malého priestorového rozsahu je preto potrebné študovať inými prostriedkami. Z toho dôvodu sa vyvíja celá škála techník, aby sa tento nedostatok odstránil. Medzi ne patria extrapolácia, štatistické metódy, empirické pravdilá a najmä konceptuálne modely.

Rozvoj v meteorológii v súčasnosti stimuluje potreba spoločnosti na kvalitné predpovede počasia na obdobie niekoľkých minút až niekoľko hodín. Takáto predpoveď, inými slovami *nowcasting*, je budovaný na podrobných aktuálnych informáciách o stave atmosféry.

Nowcasting

Striktná definícia predpovede typu nowcasting je podrobný popis súčasného počasia a jeho predpoveď na dobu do 2 hodín. Stále viac a viac sa presadzuje voľnejšia definícia nowcastingu ako predpovede, s miestnymi detailami, na dobu 0 až 6 hodín a spresnenie predpovede až do 12 hodín. [17]

Pre nowcasting je nevyhnutná vysoká frekvencia meraní stavu atmosféry pomocou automatických meteorologických staníc a dištančných meracích zariadení. Ako fundamentálne dištančné zariadenia sa používajú meteorologické rádiolokátory, meteorologické družice s celou škálou rôznych snímačov, zariadenia na meranie a lokalizáciu bleskov a profilovače vetra. Na takto získané údaje sa aplikujú špeciálne techniky, analýzy a predpovede od tých najjednoduchších (extrapolácia) až po tie najzložitejšie fyzikálne modely (konceptuálne modely).

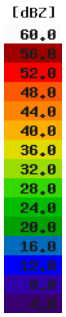
Táto aplikácia sa teda dá považovať za prostriedok, ktorý poskytuje predpoveď typu nowcasting spracovávaním aktuálnych dát z meteorologického radaru. Podobný typ aplikácie poskytuje aj Český hydrometeorologický ústav (`http://www.chmi.cz/files/portal/docs/meteo/rad/inca-cz/short.html`), no môj algoritmus výpočtu je unikátny a nija-

kým spôsobom nečerpá inšpiráciu zo spomenutého diela. Aplikácia, ktorá je výsledkom tejto bakalárskej práce čerpá údaje od ČHMÚ, ktoré sú poskytované pod licenciou Creative Commons, ktorá zakazuje zasahovať do diela. Po konzultácii s marketingovým oddelením ČHMÚ mi povolili zasahovať do radarových snímkov a výpočet úspešnosti (kapitola 5), resp. vyhodnotenie celej práce, bol konzultovaný s RNDr. Petrom Nokákom, zamestnancom ČHMÚ.

Meteorologické rádiolokátory

Meteorologické rádiolokátory slúžia na zisťovanie rozloženie okamžitých intenzít atmosferických zrážok(v diskrétnom čase) a výskytu javov spojených s oblačnosťou na veľkej ploche (do vzdialenosti približne 100 - 200km). Ich funkcia je založená na schopnosti zrážkových častíc v atmosfére (vodných kvapiek, snehových vločiek, ľadových krúp a pod.) odrážať rádiovlny v centimetrovom pásme vlnových dĺžok (mikrovlny). [14]

Štruktúra a charakter oblačného systému sú zretelné predovšetkým v priestorovom rozložení rádiolokačnej odrazivosti. Farebná stupnica radarového echa má 15 stupňov odrazivosti s krokom 4dBZ.



Obrázek 2.1: Stupnica intenzity zrážok [5]

Prahová hodnota 4dBZ odpovedá intenzite dažďa približne cca 0.06 mm/h. Pre približný prepočet odrazivosti na intenzity zrážok platí exponenciálna závislosť.

Z [dBZ]	7	23	39	55
I [mm/h]	0.1	1	10	100

Z uvedenej tabulky vyplýva že pri echu 39dBZ je intenzita zrážok 10mm/h, čo je 10 litrov dažďa na plochu $1m^2$.

Objekty pozorované radarom sa dajú deliť na dve skupiny:

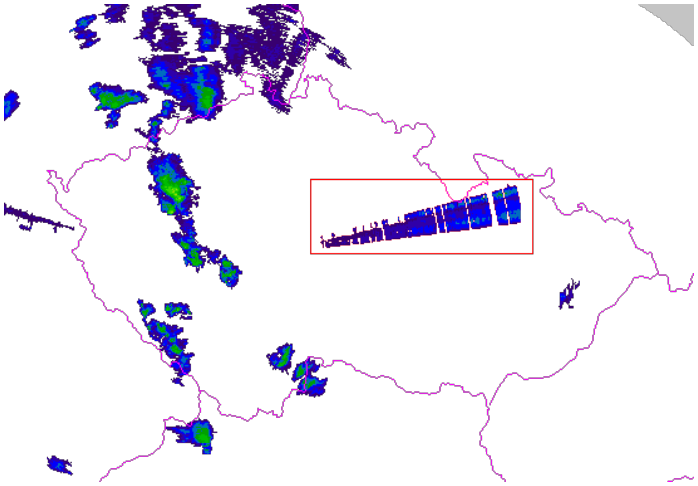
Meteorologické ciele

Tieto objekty majú obvykle väčší rozsah ako jednotlivé pixely a skôr kompaktnejší tvar. Priestorové zmeny odrazivostii sú plynulé, časové zmeny odrazivosti sú malé a na animácii sa javí zretelný pohyb. Objavujú sa a miznú postupne.

Nemeteorologické ciele

Často sa jedná o jednotlivé pixely s vyššou odrazivosťou alebo menšie nesúvislé oblasti. Pri týchto objektoch býva velká priestorová aj časová premenlivosť odrazivosti, čo znamená že majú často velmi ostré okraje a môžu sa náhle objaviť alebo zminúť. Na animácii nieje badatelný pohyb, sú statické.

Oba tieto typy objektov sú badatelné na Obr. 2.2. V červenom obdĺžniku je zobrazený nemeteorologický cieľ. Inými slovami, aj keď je na tomto území zachytené radarové echo, na zemskom povrchu nie sú registrované žiadne zrážky. Algoritmus ale nerozpoznáva tieto typy odrazov, je však vhodné uviesť tento jav a ich elimináciu prípadne zaradiť medzi možné rozšírenia projektu.



Obrázek 2.2: Ukážka rušenia - červený obdĺžnik a reálnych odrazov zrážok

2.2 Implementačné techniky

Pôvodným zámerom bolo vytvoriť natívnu aplikáciu na systéme Android, ktorá by upozorňovala pred blížiacimi sa zrážkami, komunikovala by s GPS.

2.2.1 Android OS a zvolené techniky

Android OS [2] je operačný systém od internetového giganta Google. Pre vývoj natívnych aplikácii na Android je nutné splniť niekolko požiadavkov. Základ pre vývoj je Android SDK, poskytuje API knižnice a vývojárske prostriedky nutné k úspešnému prekladu, testovaniu a debuggovaniu aplikácie.

V Androide sa programuje primárne v Jave, čo uľahčuje hlavne prvé kroky. Na druhej strane iba znalosť Javy nestačí, platí jedno zásadné pravidlo, znejúce príliš veľa abstrakcie skorej vybije baterku, je teda nutné naučiť sa programovať mobilne. Čo sa však z Javy dá využiť je znalosť všemožných API. Aplikácie sa svetu distribuujú prostredníctvom Android Marketu, členstvo je ale spoplatnené.

Od pôvodných zámerov bolo sčasti upustené a viac som sa zameral na výpočet samotnej predpovede. Vzhľadom na to, že som sa rozhodol alarm neimplementovať, nie je nutná komunikácia s GPS zariadením. Tým pádom nie je nutné aby aplikácia bola natívna, ale stačí

keď bude bežať online ako webová. Aplikácia teda bude bežať vo webovom prehliadači, čo znamená, že bude spustitelná v akomkoľvek desktopovom či mobilom zariadení. Z osobných dôvodov a záujmu som si na implementáciu zvolil nasledujúce techniky:

- backend časť - Node.js
- frontend časť - HTML, CSS, Jquery, JavaScript
- podporné technológie - Git a Heroku, Amazon S3, MongoDB, Uptime Monitor

2.2.2 Backend

Node.js [9] je vysoko výkonné, udalosťami riadené prostredie pre Javascript. Jeho základom je javascriptový interpret V8 z webového prehliadača Google Chrome, nad ním je tenká vrstva kódu v C++ poskytujúca minimálne nutné zázemie na napr. obsluhu prichádzajúcich udalostí, I/O bufferov a podobne.

V nasledujúcich odsekoch bude Node.js porovnané s inými druhmi aplikácii a jeho základná charakteristika.

Behové prostredie

Ak prijde požiadavok do webovej aplikácie, ktorá je realizovaná v PHP a webovom serveri Apache, vytvorí sa nové vlákno a oblsúži tento požiadavok. Vytvorí sa taktiež aj prostredie pre beh PHP aplikácie, nahrajú sa systémové premenné, preloží sa PHP kód a spustí sa so vstupnými dátatmi.

Aplikácia naprogramovaná v Node.js je obsahuje server priamo v sebe. Požiadavky sú teda predávané vo forme **udalostí** do zoznamu eventov, ktoré sú určené k spracovaniu. Tento zoznam je monitorovaný aplikáciou a požiadavky sú postupne obsluhované, čím nedochádza k zvýšenej réžii pri príchode viacerých požiadavkov naraz. Ďalšou výhodou je, že po spracovaní požiadavku sa neukončí celý proces, ale beží ďalej a čaká na obslúženie iných požiadavkov. V aplikácii teda môžu byť ukladané priebežné informácie, ktoré by PHP aplikácia musela pracne počítať odznova. Táto vlastnosť rieši základnú formu cachovania.

Používanie callbackov

Zjednodušene povedané, Node.js je server-side behové prostredie pre aplikácie napísané v JavaScripte. Písanie takýchto aplikácií sa nie moc líši od client-side aplikácii na strane prehliadača. Aj v Node.js aplikácii sa používajú takzvané **callbacky** skôr ako návratové hodnoty funkcii. Ukážka nižšie je priamo zo zdrojového kódu bakalárskej práce.

```
// definícia funkcie pre odoslanie json odpovede na front-end

var sendToFrontend = function ( err, frames ) {
    res.contentType( 'json' );
    res.send ( { response : frames} );
}

// funkcia na výber najnovších snímkov z databázy

mongoose.model( 'Frame' ).getLastFrames( sendToFrontend );
```

Funkcia `sendToFrontend()` je ako callback predaná funkcii, ktorá vykoná výber snímkov z databázy. To znamená, že po výbere dát z databázy nie sú predané návratovou hodnotou z funkcie, ale je zavolaná callback funkcia a tieto dáta sú jej predané ako parameter.

Event-driven a I/O non-blocking architektúra

Ako bolo spomenuté vyššie, v Node.js existuje **fronta udalostí**. Táto fronta predstavuje zoznam požiadavkov, ktoré musí server vykonať.

Najjednoduchším príkladom je požiadavok na webový server. Tento požiadavok je uložený do zoznamu udalostí a keď naňho prijde rada, dôjde k jeho spracovaniu.

Ďalším, už trochu zaujímavejším príkladom je funkcia v predchádzajúcom odseku. Tu je funkcia `getLastFrames()` požiadaná o výber dát z databázy a je jej predaná callback funkcia. Do tej doby, než prijde výsledok funkcie an výber dát z databázy, môže pokračovať obsluha ostatných požiadavkov. Táto vlastnosť je označovaná ako **I/O non-blocking** a vďaka nej môže Node.js spracovávať požiadavky asynchrónne.

Single thread

Pri vývoji Node.js bolo naschvál zvolené **jednovláknové spracovávanie** požiadavkov. Pri paralelnom spracovávaní, myslené spôsobom, že by fronta udalostí bola spracovávaná viacerými vláknami naraz, by dochádzalo k problémom so synchronizáciou, zamykaním a podobne. Preto vývojári zvolili event-driven archtektúru kde všetko prebieha v jednom vlákne. Sú však dostupné moduly, ktoré by podporili vytváranie viacerých vlákien, no to nie je záležitosť tejto práce.

Moduly

Pri programovaní v Node.js je k dispozícii množstvo vytvorených modulov. Existujú aj databázy modulov, v tejto práci bola využitá databáza modulov **NPM**, ktorá k dátumu 24.4.2014 obsahuje 69 703 modulov. Ktoré moduly boli použité v tejto práci a ich stručná charakteristika bude popísané v samostatnej kapitole neskôr.

2.2.3 Frontend

Na frontend časti boli použité iba bežné techniky, kombinácia HTML, CSS a JavaScript [12]. Čo sa týka zložitosti, je táto časť vytvorená tak, aby bola čo najjednoduchšia a užívatelsky priatelská. Všetky zaujímavé algoritmy prebiehajú na serverovej strane.

2.2.4 Podporné technológie

Pri vývoji boli použité nástroje, ktorých stručnú charakteristiku a dôvod použitia uvádza táto podkapitola.

Git a Heroku

Heroku [7] je špecializovaná cloudová platforma pre aplikácie písané v Ruby, Node.js, Clojure, Java, Python. Od bežného hostingu sa odlišuje predovšetkým tým, že sa k nemu nepristupuje cez FTP, ale pomocou nástroja Git. V základnej verzii funguje zdarma, preto je ideálnym riešením pri vývoji tejto práce.

Git [6] je mocný distribuovaný systém správy verzií a ideálny a odporúčaný nástroj pri vývoji aplikácii na Heroku. Nasleduje zoznam príkazov, ktoré boli použité / používané počas vývoja.

Inicializácia prázdneho Git repozitára:
```
$ git init
```

Nahranie aplikácie do Git repozitára:
```
$ git add .
$ git commit -m comment
```

Vytvorenie aplikácie na Heroku:
```
$ heroku create
```

Prepojenie Heroku aplikácie a Git repozitára:
```
$ heroku git:remote -a damp-sierra-6157
```

Nahranie kódu z Git repozitára a spustenie aplikácie nanovo
```
$ git push heroku master
```

Amazon Simple Storage Service

Amazon Simple Storage Service [1], vskratke Amazon S3, je internetové uložisko navrhnuté na ukladanie a obdržiavanie lubovolného množstva dát. Táto služba je využitá na ukladanie stiahnutých a vygenerovaných snímkov. Filesystem aplikácie, ktorá funguje na Heroku je iba na čítanie (read-only), čo znamená, že ak by boli tieto snímky nahraté sem, Heroku ich zmaže po určitom čase. Z praxe môžem povedať, že Heroku upratuje približne každé 2 hodiny.

Node.js a Amazon S3 spolu komunikujú skrz modul `aws-sdk`. Komunikáciu zabezpečujú 3 premenné, ktoré sú boli obdržané pri zaregistrovaní služby na Amazon.

`accessKeyId` - verejný kľúč
`secretAccessKey` - privátny kľúč
`sessionToken` - priečinok v úložisku

MongoDB

MongoDB [8] je typ open-source databázy, založená na princípoch NoSQL. Dôležitou charakteristickou črtou MongoDB je velmi dobrá prepojenosť s Node.js, pretože táto databáza príjma dokumenty vo formáte JSON (JavaScript Object Notation). Použitá databáza je rozšírenie, ktoré poskytoval Heroku. Do databázy sú ukladané informácie o jednotlivých radarových snímkoch a úspešnosť predpovede.

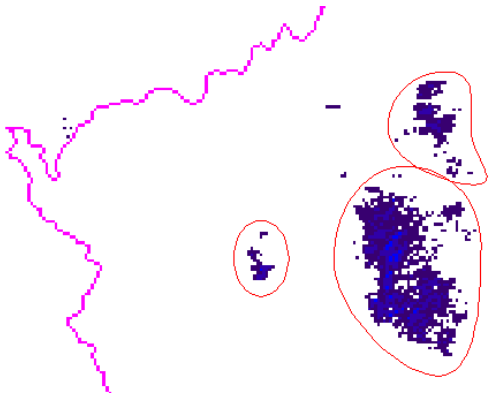
Uptime Monitor

Uptime Monitor [11] je ďalšia z radu free utilít, ktorá monitoruje dostupnosť aplikácie. Monitorovanie však nie je primárny dôvod jej použitia. Ak je aplikácia na Heroku nečinná približne hodinu (myslená užívatelská činnosť), systém ju uspí. To znamená že aktualizácia

snímkov je zastavená a tomu som chcel predísť. Tento monitor sa opakovane dotazuje na aplikáciu každých 5 minút, čím vykonáva aktivitu a systém aplikáciu neuspáva.

2.3 Prierez zhlukovacími metódami

Zhlukovacie metódy [20] tvoria jednu z hlavných častí zadania a v samotnom výpočte predpovede je použitá jedna konkrétna metóda. Uvediem prehľad zhlukovacích metód.



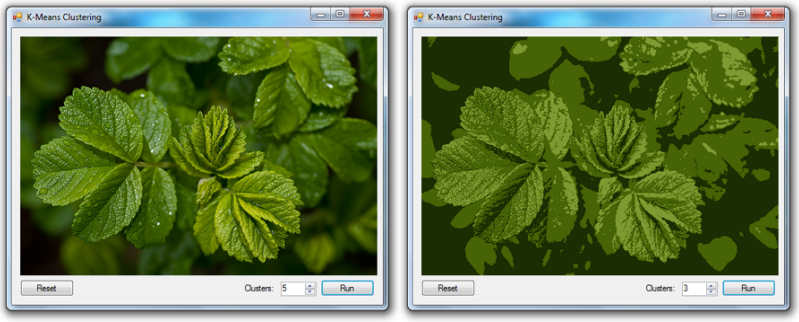
Obrázek 2.3: Ukážka stiahnutého radarového snímku

Bitmapu stiahnutú z webu ČHMÚ odzrkladluje obrázok 2.3. V ideálnom prípade bude nasledujúci záber rozdelený do dvoch / troch zhlukov. Zvyšok bude ignorovaný (ako šum), alebo spojený do jedného vačšieho zhluku.

Metódy založené na rozdeľovaní

Patria sem metódy K-means, K-medoids. Pri týchto metódach sa počet zhlukov zadáva hneď na začiatku analýzy. Tento algoritmus pracuje tak, že každý objekt priradí do triedy ku ktorej stredu je najbližšie. Stredy zhlukov sa spočítavajú vždy pri behu algoritmu ako aritmetický priemer všetkých bodov zhluku.

Pri práci s pixelmi obrázku sa dá použiť napríklad na segmentáciu obrázka, príklad je uvedený na nasledujúcom obrázku. Vľavo je pôvodný obrázok, vpravo je obrázok rozdelený na segmenty podla farieb. Na využite v mojej aplikácii ale vyzerá byť tento spôsob zhlukovania nevyužitelný.



Obrázek 2.4: Využitelnosť zhlukovacej metódy K-means [19]

Hierarchické metódy

Hierachické metódy (algoritmy Diana, Agnes...) vytvárajú hierarchický rozklad danej množiny objektov, pričom vzniká strom zhlukov. Pri práci s pixelmi ma ale nenapadá žiadne rozumné použie, zato omnoho použitelnejšie vyzerá nasledujúca skupina metód založených na hustote.

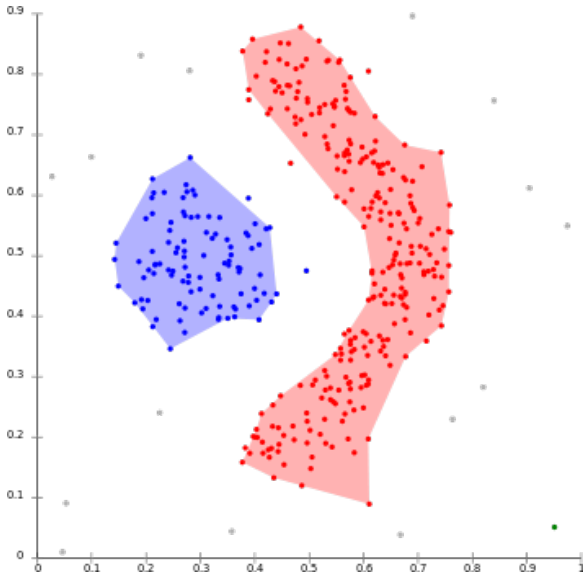
Metódy založené na hustote

Algoritmy - DBSCAN, DenClue. Tieto algoritmy považujú za zhluky oblasti s velkou hustotou objektov, v priestore dát ktoré sú od seba oddelené oblasťami s malou hustotou vyskytujúcich sa objektov. Objekty, ktoré sa vyskytujú v oblasti s malou hustotou objektou sú považované za šum.

Tieto metódy sú nevyužitelné pri rovnomerne rozložených objektoch, to mi ale neprekáža. U radarového snímku budú existujúce zhluky krásne oddelitelné od zvyšku. Tieto algoritmy si dokážu poradiť so šumom a dokážu identifikovať zhluky rôzneho tvaru, čo sa taktiež velmi hodí.

Demoštrácia práce algoritmu DBSCAN, ktorá by sa dala využiť aj pri práci s pixelmi je uvedená na Obr. 2.5.

Metóda DenClue pracuje na matematickom základe, využíva distribučnú funkciu.



Obrázek 2.5: Využitelnosť zhlukovacej metódy DBScan [4]

Je postavená na myšlienkach:

- Vplyv každého objektu môžeme formálne modelovať pomocou matematickej funkcie - funkcie vplyvu, ktorá zachytáva vplyv objektu v jeho okolí.
- Celkovú funkciu hustoty dátového priestoru môžeme modelovať analyticky ako súčet jednotlivých funkcii vplyvov všetkých objektov v priestore.
- Zhluky potom sú matematicky určené miesta v priestore, kde sa nachádzajú lokálne maximá celkovej funkcie hustoty.

Metódy založené na mriežke

Algoritmus - WaveCluster, rozdeluje dátový priestor pomocou mriežky. Každá bunka zahŕňa informácie o skupine objektov, ktoré sú do nej namapované. Následne sa na hodnoty v mriežke použije viacúrovňová vlnková transformácia. Táto metóda taktiež vyzerá použitelne.

Metódy založené na modeloch a metódy pre zhlukovanie vysoko-dimenzionálnch dát

Metódy založené na modeloch sú pre môj prípad príliš komplikované, nakolko mne stačí identifikovať zrížkové polia a nepotrebujem ich ďalej porovnávať s nejakým matematickým modelom. Metódy pre zhlukovanie vysoko-dimenzionálnch dát sú uplne mimo moju problematiku.

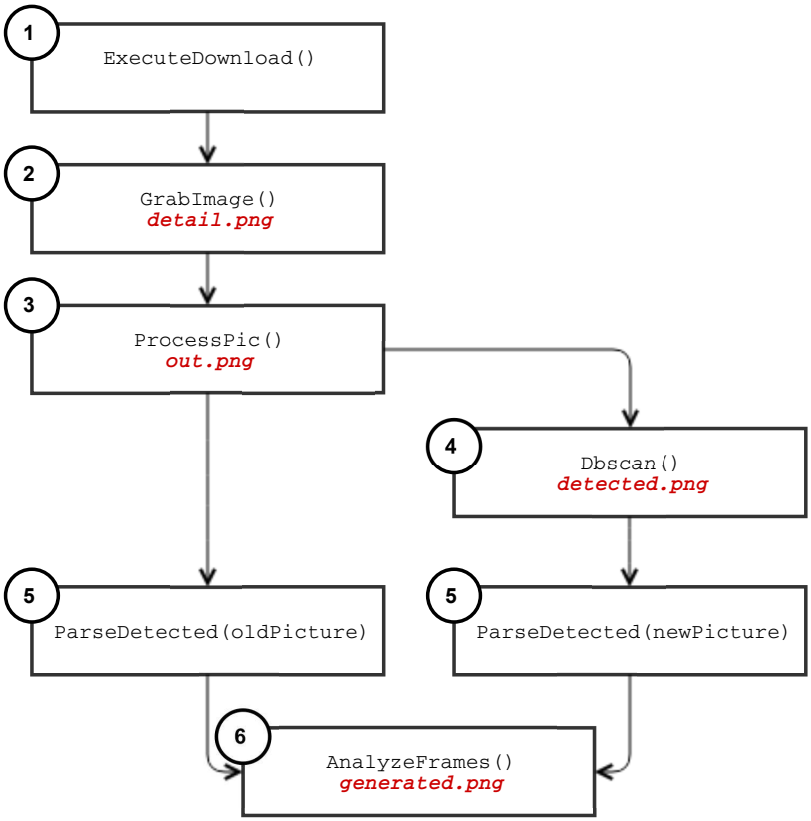
Kapitola 3

Vytvorenie predpovede

Vytvorenie predpovede spočíva na troch hlavných pilieroch:

- ***1. stiahnutie snímku***
- ***2. spracovanie najnovšieho a predchádzajúceho snímku***
- ***3. zo spracovaných dát vygenerovanie predpovede***

Algoritmus výpočtu je ilustrovaný na Obr. 3.1.



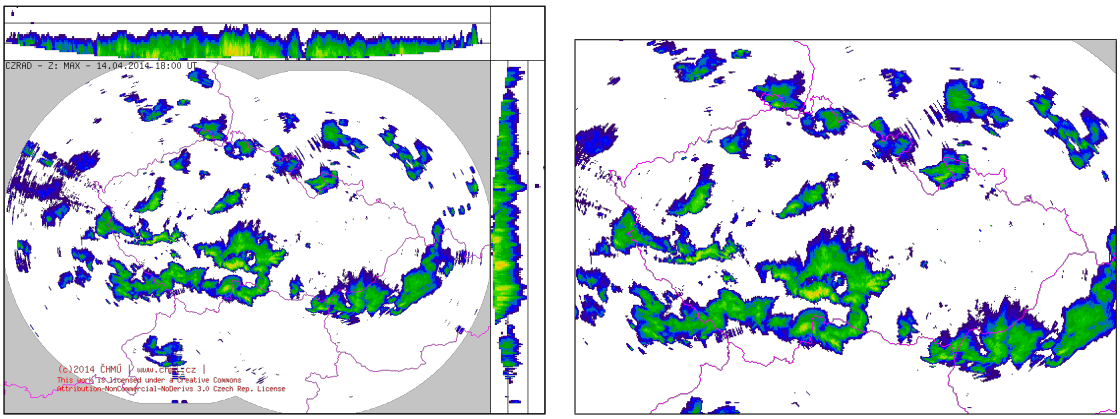
Obrázek 3.1: Výpočet v kocke

Jednotlivé kroky sú označené číslami tak ako vo výpočte nasledujú za sebou. Najprv je potrebné identifikovať najnovší snímok - funkcia `ExecuteDownload()`. Nasleduje funkcia

`GrabImage()`, ktorá stiahne tento nájdený snímok a ten uloží ako ***detailXY.png***. Nasleduje spracovanie snímku ako orezanie, dekódovanie - funkcia `ProcessPic()`, jej výstupom je spracovaný snímok ***outXY.png***. Ďalej je snímok spracovávaný výpočtami, ktoré predstavuje funkcia `Dbscan()`. Výsledkom týchto výpočtov je ***detectedXY.png***, čo je spracovaný snímok v konečnej podobe. Funkcia `ParseDetected()` počká kým sú pripravené dáta z predchádzajúcich výpočtov, tieto dáta prevezme a predá ďalej funkcii `AnalyzeFrames()`, ktorá vygeneruje predpovedný snímok ***generatedXY.png***.

V aplikácii sa teda pracuje so štyrmi druhmi snímkov. 3 z nich sú priamo viditeľné v GUI aplikácie, jeden je pôvodný, z ktorého vzniknú ostatné.

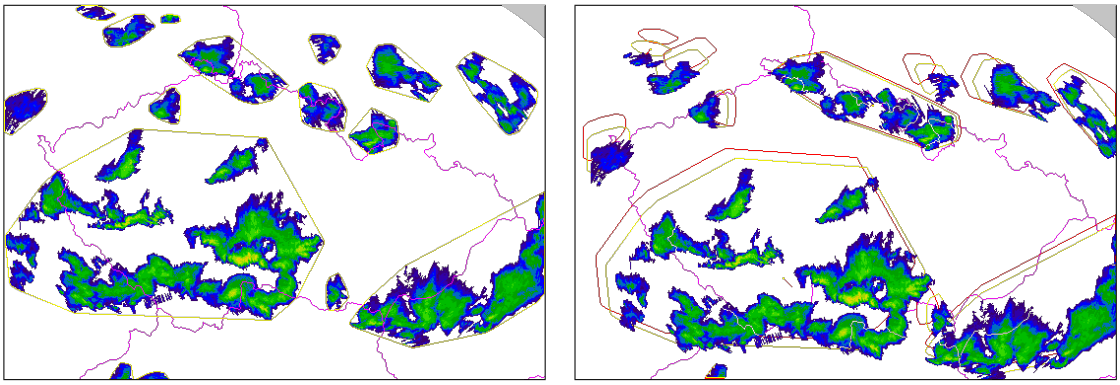
- **detailXY.png** - pôvodný snímok, needitovaný, o rozmeroch 810 x 610 pixelov, Obr. 3.2 vľavo
- **outXY.png** - orezaný snímok, o rozmeroch 550 x 380 pixelov, s takýmto rozmerom sa ďalej pracuje, Obr. 3.2 vpravo
- **detectedXY.png** - snímok na ktorom sú odstránené pixely, ktoré boli identifikované ako šum, a žltou farbou sú označené zrážkové zhluky, Obr. 3.3 vľavo
- **generatedXY.png** - predpovedný snímok, červenou farbou sú označené staršie zhluky zrážok, žltou novšie, obsahuje taktiež ťažiská zhlukov aj s vektormi posunu, Obr. 3.3 vpravo



Obrázek 3.2: detailXY.png (vľavo) a outXY.png (vpravo)

Dvojica znakov XY v názvoch snímkov predstavuje číselnú dvojicu podla času, kedy bol snímok stiahnutý. Snímky sú na webe ČHMÚ aktualizované po 15 minútach (o 0:00, 0:15, 0:30...), čo znamená, že za predpokladu nenastania chyby radaru, maximálne 96 snímkov denne. Snímok s názvami ***detail4.png***, ***out4.png***, ***detected4.png*** prislúcha času 1:00.

So snímkom generatedXY je to už troška komplikovanejšie, pretože každých 15 minút sú generované 4 predpovedné snímky. To znamená maximálne 384 predpovedných snímkov denne. Dvojica XY pri snímku generated predstavuje XY v ostatných názvoch snímkoch prislúchajúcim danému času, vynásobená číslom 4. Teda v čase 1:00 budú generované predpovede s názvami ***generated16.png***, ***generated17.png***, ***generated18.png***, ***generated19.png*** (predpoveď na 15, 30, 45 a 60 minút).



Obrázek 3.3: detectedXY.png (vľavo) a generatedXY.png (vpravo)

3.1 Krok 1 - Získanie radarových snímkov

Vstupom do prvého kroku je štart aplikácie, žiadne dáta. Výstupom je stiahnutý radarový snímok.

V hlavnom súbore aplikácie je nastavený timer, ktorý spúšťa operáciu na stiahnutie najnovšieho snímku, s časovým odstupom 15 minút. Radarový snímok je dostupný na adrese vo formáte:

`http://www.chmi.cz/files/.../pacz23.z_max3d.RRRRMMDD.HHMM.0.png`

Kde RRRR je rok, MM mesiac, DD deň, HH hodina a MM minúta. URI cesta je týmto pádom ku každému snímku iná a aplikácia musí nájsť ten najaktuálnejší. Z akutálneho času sa získa približný čas najnovšieho snímku a odošle sa požiadavok na stiahnutie na web ČHMÚ. Od aktuálneho času sú odrátané 2 hodiny, keďže na webe ČHMÚ sa pracuje s časovým formátom UTC - Coordinated Universal Time. Pri zimnom čase na našom území je čas rovnaký ako UTC - 1 hodina, pri letom čase je rozdiel dvojhodinový.

Ak je odpoveď `200 OK`, znamená, že snímok existuje, ale aplikácia nevie či je to najaktuálnejší. Preto inkrementuje hodnotu času o 15 minút a požiadavok sa odošle znova. Tento postup sa opakuje, až pokial odpoveď od servera nie je `404 Not Found`. Vtedy je zrejmé, že novší snímok už neexistuje a najnovší bude ten posledný s kladnou odpoveďou.

Snímok je stiahnutý, ďalej spracovávaný a uložený do Amazon S3 - Simple Storage Service. Do databázy MongoDB sú uložené údaje o adrese k snímku (smerujú do Amazon úložiska) a čas vo fromáte HHMM a dátum vo formáte RRRRMMDD.

Pre tento účel bol použitý modul `Request`.

3.2 Krok 2 - Orezanie snímku

Vstupom do druhého kroku je snímok stiahnutý v prvom kroku, výstupom je orezaný snímok o rozmeroch 550px a 380px.

Stiahnutý snímok je obrázok vo formáte `.png` a preto je tento obrázok potreba načítať do nejakej štruktúry s ktorou sa dá ďalej pracovať. Tento proces zastrešuje ďalší použitý modul `PNGJS`. Asynchrónne zavolaná hlavná funkcia z tohto modulu načíta predložený PNG obrázok do jednorozmerného poľa, kde o každom pixeli uchováva 4 hodnoty RGBA - red, green, blue, alpha. Pôvodný obrázok má rozmery 810 x 610 pixelov, pole má teda 494 100

prvkov. Pre ďalšiu prácu je pole pretransformované do dvoch rozmerov = maticu o 810 stĺpcov a 610 riadkov. Pri dekódovaní sa uchovávajú o každom pixeli aj ďalšie užitočné informácie - 4 hodnoty RGBA, x-ová súradnica, y-ová súradnica a váha pixelu, ktorá záleží na intenzite zrážok v tomto pixeli - hodnota od 0 do 15, kde 15 je maximálna intenzita, 0 znamená že v tomto mieste neboli identifikované žiadne zrážky (Obr. 2.1).

Na dekódovaný obrázok je aplikovaná funkcia `cropMatrix` ktorá oreže dvojrozmerné pole podla rozmerov, ktoré sú jej predané prostredníctvom parametrov. Radarový snímok je upravený tak, aby Česká republika bola v snímku celá a okolité územia sú v primeranej miere orezané. Výsledný snímok má rozmedy 550 x 380 pixelov, čím sa pôvodné pole zmenší na 209 000 prvkov. Dvojrozmerné pole je pretransformované naspäť na jednorozmerné a zakódované do výstupného obrázku outXX.png za pužitia funkcie z modulu `PNGJS`. Tento obrázok je uložený, tak isto ako detailXX.png z predchádzajúceho kroku, na Amazon S3.

3.3 Krok 3 - Identifikácia zhlukov na snímku

Vstupom do tretieho kroku je orezaný snímok outXX.png z predchádzajúceho kroku. Výstupom je pole zhlukov (bude ilustované na obrázku).

Predložené je dvojrozmerné pole o rozmeroch 550 x 380, pochádzajúce z minulého kroku. Na toto pole je aplikovaná zhlukovacia metóda.

Vybraný algoritmus - DBSCAN

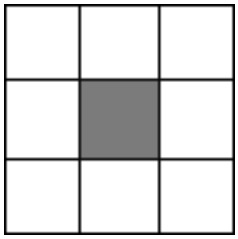
Na identifikáciu zrážkových zhlukov som si vybral algoritmus DBSCAN [16], veľmi dobre pasuje k mojej problematike. Algoritmus postupne prechádza jednotlivé prvky a keď narazí na nejaký prvok, ktorý nepatrí do nijakého zhluku, začne ho expandovať. Hladá v okolí podľa zadaného parametra ϵ. Ak sa mu nepodarí nájsť už žiadne ďalšie prvky, ktoré by mohol zahrnúť do zhluku, je daná štruktúra prvkov identifikovaná ako samostatný zhluk. V závere ešte porovná počet prvkov v štruktúre s hraničným parametrom, ak je počet menší ako tento parameter, je zhluk zahodený - identifikovaný ako šum.

Pseudokód algoritmu DBSCAN

```
dbscan(Matrix) {
    Clusters = [];
    Foreach Pixel in Matrix {
        if ( ! Pixel.visited ) {
            if ( isPrecipitationPixel( Pixel ) ) {
                NeighborPoints = getNeighborPts( Pixel.x, Pixel.y );
                NewCluster = expandCluster( NeighborPoints );
                Clusters.push( NewCluster );
            }
        }
    }
}

expandCluster(NeighborPts) {
    Cluster = [];
    foreach Pixel in NeighborPts {
        if ( ! Pixel.visited ) {
```

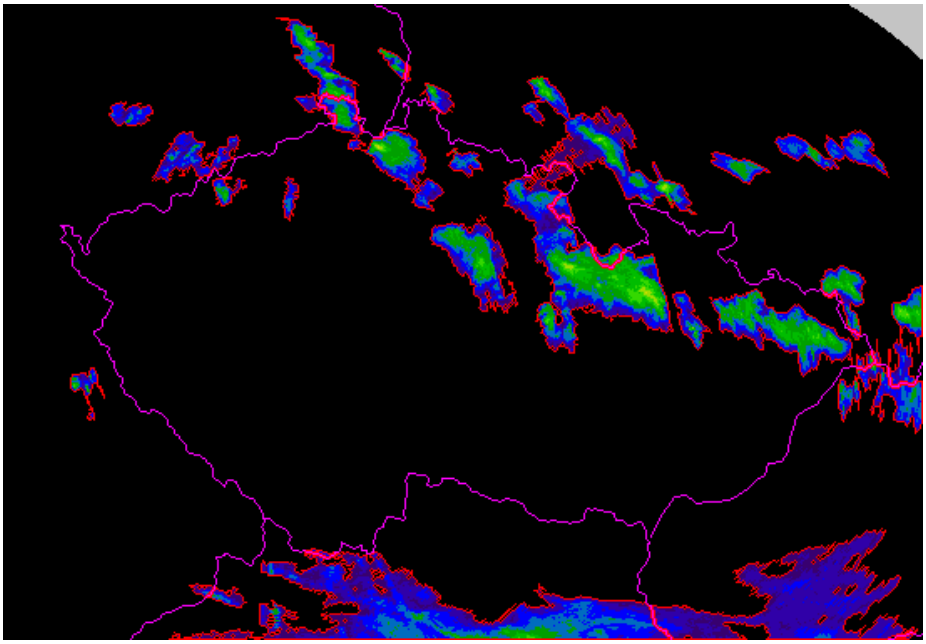
```
                Pixel.visited = true;
                NewNeighborPts = getNeighborPts( Pixel.x, Pixel.y );
                NeighborPts.push( NewNeighborPts );
                Cluster.push( Pixel );
            }
        }

        return Cluster;
    }
```

Funkcia `getNeighborPts( x, y )` vracia pole pixelov, ktoré sú identifikované ako zrážkové a sleduje najbližšie okolie pixelu do vzdialenosti 1, ilustrácia na Obr. 3.4.



Obrázek 3.4: Získanie susedných zrážkových pixelov na všetky svetové strany

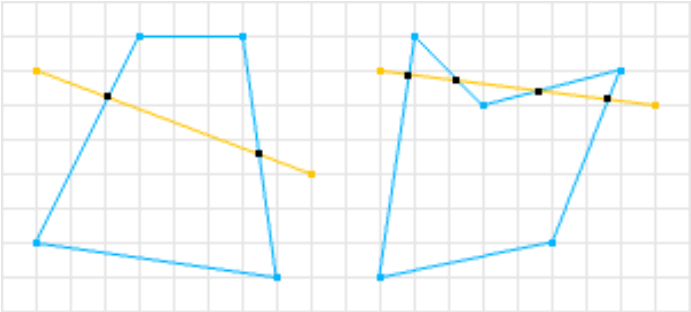
Nakoniec sú odstránené zhluky, ktoré obsahujú menej ako 150 pixelov, tieto sú označené ako šum radaru a nemá význam spracovávať takto malé zhluky. Ako sú identifikované zhluky na kompletne celom radarovom snímku zobrazuje Obr. 3.5, identifikované zhluky sú obtiahnuté červenou farbou.



Obrázek 3.5: Identifikované zhluky zrážok obtiahnuté červenou

3.4 Krok 4 - Spracovanie zrážkových zhlukov

Vstupom do štvrtého kroku je pole zhlukov predchádzajúceho kroku. Výstupom je pole konvexných polygonov (bude ilustované na obrázku).



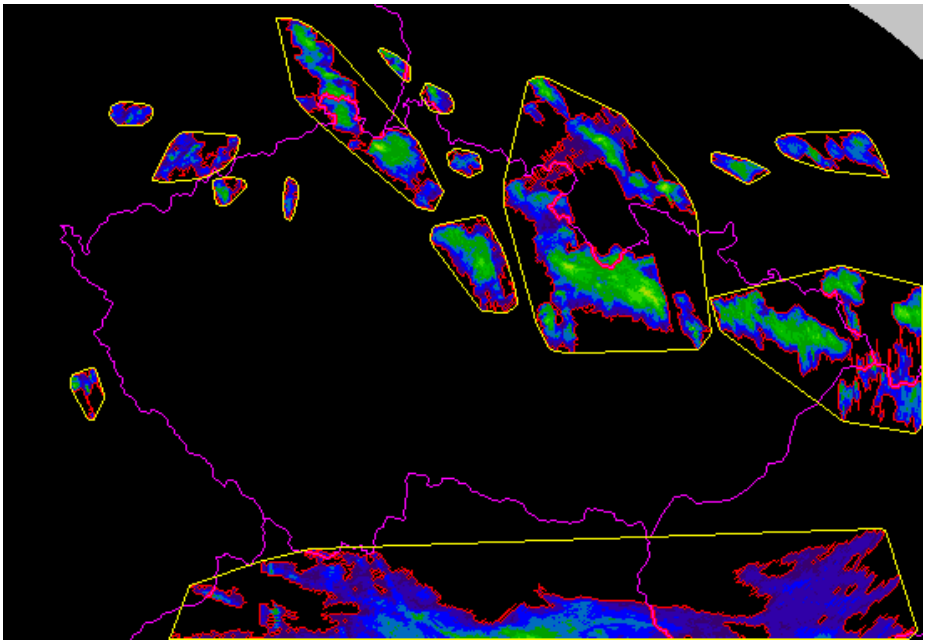
Obrázek 3.6: konvexný polygon - vľavo vs. nekonvexný polygon - vpravo [10]

Rovinný mnohouholník nazývame konvexným - vypuklým, ak pre každú z jeho strán platí, že ostatné vrcholy ležia v jednej polrovine, určenej priamkou obsahujúcou túto stranu [13] . Na výpočet nosných bodov konvexného mnohouholníka bol zvolený algoritmus Monotone Chain od pána Andrewsa [15].

Tento algoritmus postupne prechádza pole prvkov dvakrát. Pri prvom prechode hľadá hraničné body spodnej polovice zhluku, pri druhom prechode zase hraničné prvky hornej polovice. Nakoniec sú hraničné body uložené do poľa a výsledkom je niekoľko bodov identifikujúcich konvexný polygon, ktorý nahrádza pri ďalšej práci zrážkový zhluk (mnohokrát väčšia štruktúra bodov). Pseudoalgoritmus je uvedený nižšie a výsledok tohto kroku je ilustrovaný na Obr. 3.7.

```
convexHull(Cluster){
    Cluster = Cluster.sortByCoordinate(x);

    lowerHull = [];
    upperHull = [];

    forEach Point in Cluster{
        while ( lowerHull.length >= 2 &&
                isRightTurn( lowerHull[-2], lowerHull[-1], Point ) ){
            lowerHull.pop( );
        }
        lowerHull.push(point);
    }

    forEach Point in reverseCluster{
        while ( upperHull.length >= 2 &&
                isRightTurn( upperHull[-2], upperHull[-1], Point ) ){
            upperHull.pop( );
        }
        upperHull.push( point );
    }
```

```
    convexHull = concat( lowerHull, upperHull );
}
```



Obrázek 3.7: Identifikované konvexné polygóny znázornené žltou

3.5 Krok 5 - Výpočet centroidov a identifikácia súvisiacich zhlukov

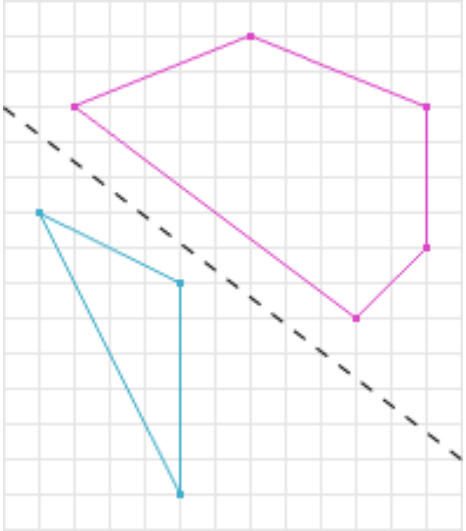
Vstupom do piateho kroku sú dve polia konvexných polygonov a dve polia zrážkových zhlukov. Dve (z najnovšieho snímku a z druhého najnovšieho snímku) preto aby bola identifikovaná súvislosť medzi zhlukmi zrážok na dvoch nezávislých snímkoch.

Algoritmus porovnáva vytvorené polygony a ak dojde medzi dvoma polygonmi k prieniku, zlúči ich do jedného a výpočet jeho nosných bodov sa musí opakovať. Takto dojde k zlúčeniu zrážkových zhlukov, ktoré priamo spolu nesusedia, ale sú blízko seba a je možné ich považovať za celok.

Na detekciu súvisiacich polygonov, ako aj vrámci jedného snímku aj na identifikáciu pohybujúceho sa zhluku zrážok na rôznych snímkoch, bol použitý algoritmus SAT - Separating Axis Theorem [10].

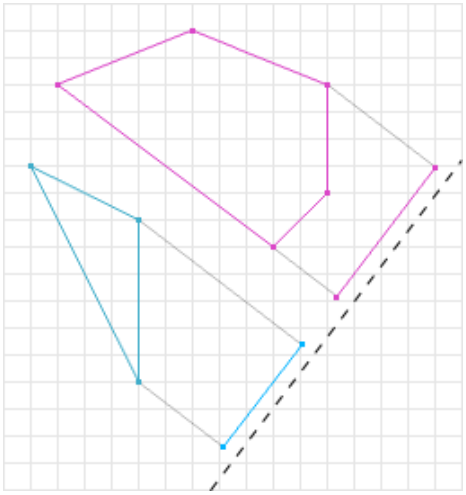
Separating Axis Theorem, vskratke SAT, je metóda na zistenie prieniku dvoch konvexných mnohouholníkov. Algoritmus môže byť využitý aj na nájdenie minimálneho prienikového vektoru, ktorý sa dá ďalej dobre využiť pre fyzikálne simulácie. Algoritmus spočíva v myšlienke, že ak sa konvexné mnohouholníky do seba navzájom neprenikajú, musí medzi nimi existovať priamka, ktorá nepretína ani jeden z porovnávaných mnohouholníkov. A túto priamku agoritmus hladá.

Ďalším konceptom, ktorý SAT algoritmus využíva je projekcia. Predstaviť sa dá ako situácia, keď svetelné lúče vychádzajúce z nejakého svetelného zdroja dopadajú na 3D objekt



Obrázek 3.8: Hlavná myšlienka SAT algoritmu [10]

pričom vytvárajú 2D tieň. Ak uvažujeme lúče z rovnakého svetelného zdroja dopadajúce na 2D objekt (náš prípad) vytvárajú jedno-dimenzionálny tieň. Situáciu ilustuje Obr. 3.9.

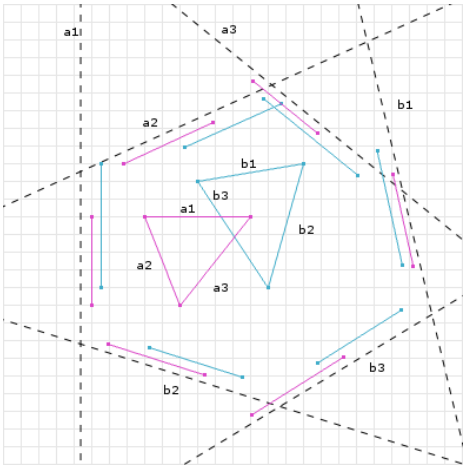


Obrázek 3.9: Ukážka projekcie [10]

Agloritmus takýmto spôsobom testuje niekolko ôs, konieckoncov ak nájde nejakú dvojicu neprelínajúcich sa tieňov, končí a vyhodnotí dvojicu polygonov ako neprelínajúcu sa. Konečný výpočet prieniku dvoch mnohouholníkov ilustruje Obr. 3.10.

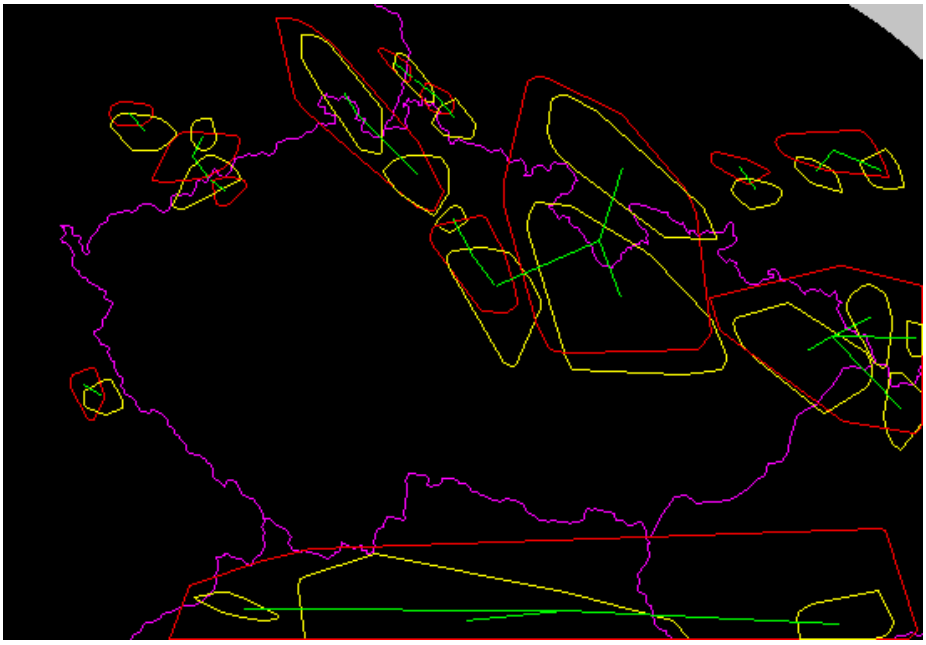
Centroidy sú vypočítané na základe váhy jednotlivých pixelov zrážkového poľa. Polda webu ČHMÚ je 15 základnýh intenzít zrážok (Obr. 2.1). Pri dekódovaní smínku v prvých krokoch dochádza aj k ohodnoteniu daného pixelu hodnotou od 0 do 15 na základe tejto stupnice. Pixel teda obshuje jednu z hexa kombinácii stupnice, alebo je jeho váha 0. Pri výpočte centroidu sa spočítavajú váhy jednotlivých pixelov zhluku a sú podelené dĺžkou, respektíve šírkou.

Ďalej je aplikovaný algoritmus, použitý aj v predchádzajúcom kroku, pre identifikáciu pri-



Obrázek 3.10: SAT algoritmus pri detekcii prieniku dvoch trojuholníkov [10]

eniku medzi polygonmi. Tentokrát ale nie medzi polygonmi vrámci jedného snímku, ale porovnávané sú polygóny na dvoch najnovších snímkoch. Výsledok je štruktúra v ktorej sú uložené zhluky ktoré sú identifikované ako ten istý zhluk no s inou polohou. Obrázok 3.11 ukazuje polygony z dvoch najnovších snímkov. Červené polygony sú staršie, žlté sú novšie. To znamená, že z červených polygonov znikli žlté. Zelenými čiarami sú spojené ťažiská zhlukov, ktoré sa prelínajú. Z týchto dát budú spočítané vektory pohybu jednotlivých zrážkových polí.

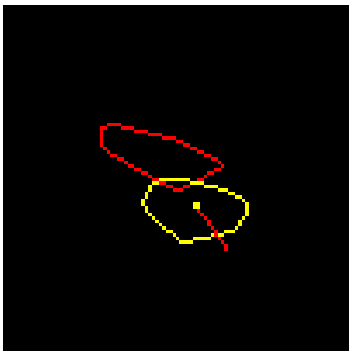


Obrázek 3.11: Polygony z dvoch snímkov a identifikácia súvisiacich polygonov

3.6 Krok 6 - Výpočet vektorov pohybu a vygenerovanie predpovede

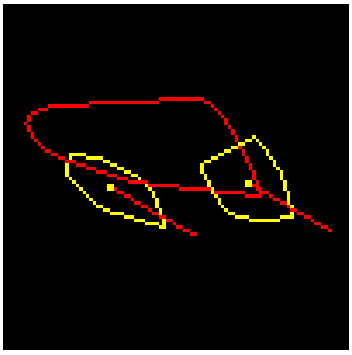
Vstupom do šiesteho kroku sú dáta z kroku piateho a výstupom je už vygenerovaný snímok.

Najjednoduchší prípad pre výpočet vektoru pohybu je vtedy, keď zhluk sa zmení minimálne a iba sa posunie. Vtedy nedochádza k rozdeleniu zhluku, k rozšíreniu zrážkového pola a podobne. Vtedy je vektorom pohybu vektor medzi centroidmi tohoto zhluku na dvoch po sebe nasledujúcich snímkoch a aplikovaný do predpovedného snímku (Obr. 3.12).



Obrázek 3.12: Ideálny prípad pohybu zhluku a vektor pohybu aplikovaný do predpovede

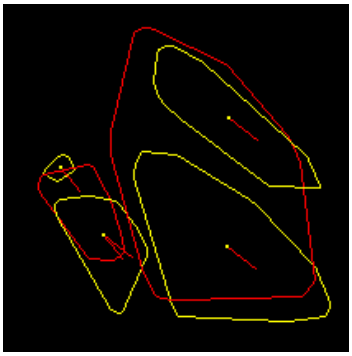
Tento ideálny prípad ale nenastáva vždy. Môžu nastať 2 nasledujúce komplikovanejšie prípady. Napríklad dojde k rozdeleniu zlhuku na x menších zhlukov (Obr. 3.13). V tomto prípade sú pôvodné vektory spočítané (vektorový súčet) a výsledný vektor je vydelený počtom zhlukov, na ktorý sa pôvodný zhluk rozdelil. V tomto konkrétnom prípade bude vektor vydelený hodnotou dva.



Obrázek 3.13: Rozdelenie zhluku na 2 menšie

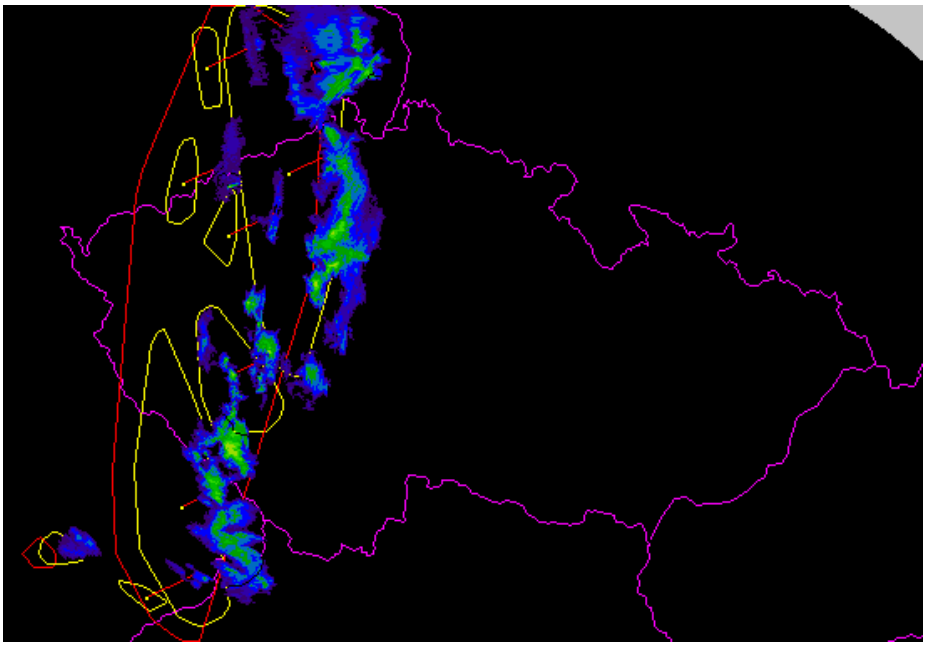
Najkomplikovanejší prípad nastáva keď sa zhluky zlúčia do jedného (Obr. 3.14). V tomto prípade bývajú výsledné vektory pohybu najmenej presné, pretože sa stáva, že sa prelínajú aj zhluky, ktoré spolu nesúvisia.

Vylepšením, ktoré pomáha k lepšiemu výpočtu vektorov pohybu sú koeficienty, ktorými sú jednotlivé vektory pohybu násobené. Tie sú vypočítané ako pomer pixelov nachádzajúcich sa v zrážkových zhlukoch. To znamená že zrážkový zhluk, ktorý zaberá na dvoch snímkoch najvačšiu plochu má vektor pohybu s najvačšou prioritou. Čím je podiel pixelov v identifikovaných zhlukoch menší, tým je aj vypočítaný vektor pohybu vynásobený menším koeficientom.



Obrázek 3.14: Vlavo dole je zhluk, ktorý legicky nesúvisí s identifikovaným zhlukom

Nakoniec je pole vektorov pohybu aplikované na zhluky najnošieho snímku, tie sú posunuté a vykreslené do prázdneho snímku (Obr. 3.15).



Obrázek 3.15: Vygenerovaný snímok

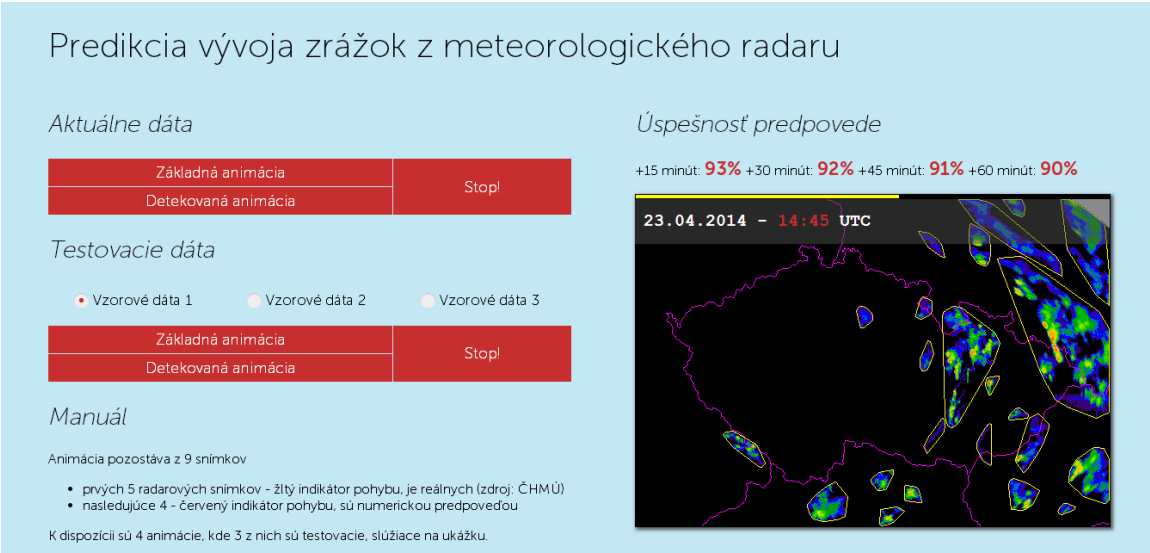
Tento algoritmus je následne opakovane použitý tak aby boli vytvorené 4 snímky, ktoré predstavujú predpoveď na 15, 30, 45 a 60 minút. Predpoveď na 15 minút je vypočítaná z dvoch najnovších reálnych snímkov, predpoveď na 30 minút je vypočítaná z najnovšieho snímku a vygenerovanej predpovede na 15 minút. Snímok na +60 minút je vypočítaný z predpovedaných snímkov na 30 a 45 minút.

3.7 Krok 7 - Front End

Samotný výpočet je naprogramovaný v javaScripte a kompletne prebieha na serveri. Aplikácia, ktorá sprostredkúva informácie užívateľovi nevykonáva žiadnu časť výpočtu. Pri otvorení aplikácie je skrz asynchrónny javascript zavolaná funkcia, ktorá z databázy získa údaje o tom, ktoré snímky by sa mali v animácii objaviť. Ak je aplikácia zavolaná akurát v

okamihu, že výpočet prebieha, snímky pochopitelne nie sú načítané a animácia je funkčná iba so snímkami, ktorých výpočet bol ukončený. Snímky sú teda načítané ešte predtým ako je spustená animácia. Spustená animácia následne funguje spôsobom, že javascriptový timer použije na jeden snímok funkciu `FadeIn()` a na jeden `FadeOut()`. Týmto je docielený decentný efekt prepínania medzi snímkami, pre lepšiu predstavivosť odporúčam otvoriť samotnú aplikáciu. (http://damp-sierra-6157.herokuapp.com/)

K aplikácii boli taktiež pridané 3 vzorové animácie, slúžiace na ukážku. Vytvorené boli pôvodne kvôli účelu, že počas programovania jadra výpočtového algoritmu v Českej republike panovalo velmi pekné bezzrážkové počasie. Nakoniec boli ponechané, pretože dobre odzrkadlujú výpočet. Úspešnosti predpovedí a vyhodnotením úspešnosti bude venovaná samostatná kapitola.



Obrázek 3.16: Screenshot aplikácie

Kapitola 4

Implementácia

Organizácia celej aplikácie vyzerá nasledovne:

```
|   app.js
|   package.json
|
+---config
|       config-dev.js
|       config-prod.js
|
+---libs
|       grab.js
|       matrix.js
|       pic.js
|       print.js
|
+---node_modules
|
+---public
|   +---images
|   +---javascripts
|   \---stylesheets
|
+---routes
|       index.js
|
+---schema
|       frame.js
|       percentage.js
|
+---views
        index.hjs
```

`App.js` tvorí jadro celej aplikácie, prebieha tam inicializácia spojenia s databázou a celého prostredia. Obsahuje aj hlavný timer, ktorý zabezpečuje aktualizáciu dát.

Priečinok `config` obsahuje konfigurácie pre produkčné prostredie a vývojové prostredie.

V priečinku `libs` sa nachádzajú hlavné časti algoritmu na sťahovanie snímku z webu - `grab.js`, maticové operácie - `matrix.js`, výpočty so snímkom - `pic.js`, vykreslovanie - `print.js`.

V priečinku `public` sa nachádzajú dáta použité na front-ende aplikácie ako obrázky, javascripty a súbory s kaskádovými štýlmi.

`Routes` obsahuje ovládacie súbory pre jednotlivé podstránky celej webovej aplikácie. Bakalárska aplikácia je typu single-page, takže priečinok obsahuje iba jeden základný súbor.

`Schema` zahŕňa súbory, ktoré predstavujú formát dát, s ktorým sa pracuje na databázovej úrovni. V databázi sú uložené údaje o snímkoch a ešte úspešnosť. Tieto dáta predstavujú súbory `frame.js` a `percentage.js`.

Posledným priečinkom je `index.hjs` čo je gui aplikácie, ktoré je viditelné v prehliadači vo formáte Hoganjs.

4.1 Použité Node.js moduly

Medzi moduly, ktoré som neprogramoval ja, ale sú volne dostupné na internete a ich inštalácia prebieha príkazom nmp patria

AWS SDK

Modul, zastrešujúci vývoj aplikácii / knižníc, ktoré využívajú služieb AWS (Amazon Web Services). Poskytuje jednoduché API pre využitie v prehliadači, tak isto aj priamo v Node.js aplikácii. V mojej aplikácii nachádza využitie pri komunikácii s Amazon S3 serverom, kde sú uložené radarové snímky.

Express

Express je minimálny a flexibilný Node.js framework. Poskytuje množstvo nástrojov pre vývoj single-page, multi-page alebo hybridných webových aplikácii. Utvára logickú aj fyzickú štruktúru celej aplikácie.

Hjs

Hogan.js (Hjs) je prekladač Mustage šablonového jazyka. Jedná sa o obdobu HTML, do ktorého sú predané premenné priamo z back-endu a rôznych skriptovacích konštrukcii. Takýmto spôsobom je napísaný front-end aplikácie.

Less Middleware

Tento modul je prekladač určený pre CSS. Základné CSS rozširuje o radu rôznych možností. V práci je použité, ale CSS je samo o sebe minimálna časť, preto je využité minimálne množstvo možností.

Long John

Tento modul poskytuje sledovanie zásobníkov s konfiguratelnou hĺbkou. Využitie nachádzal pri vývoji, pri oprave chýb.

Mkdirp

Modul, ktorý poskytuje funkcie na prácu s priečinkami. Má podobné funkciu ako unixový príkaz `mkdir -p`. Využitý je pri ukladaní snímkov, kde treba snímky z každého dňa uložiť do samostatného priečinka a tento priečinok treba nejakým spôsobom najprv vytvoriť.

Mongoose

Mongoose modul poskytuje priame, na schéme založené riešenie modelovania dát a komunikáciu s databázou MongoDB. V aplikácii sú použité dve schémy, prvou je model snímku, kde sú ukladané informácie o snímku - dátum, čas, URI adresa smerujúca na Amazon server. Druhou schémou je model percentuálneho vyjadrenia úspešnosti predpovede.

Nanobar

Veľmi lightweight progress bar podporovaný všetkými aktuálnymi prehliadačmi, bez použitia jQuery. Použitý pri animácii snímkov na front-ende.

NewRelic

Monitor Node.js aplikácie, poskytujúci GUI webové rozhranie s množstvom ukazovatelov o aplikácii.

Pngjs

Jednoduchý kóder / dekóder obrázkov vo formáte PNG. Použitý je niekolkokrát pri zakódovaní a dekódovaní snímkov.

Request

Poskytuje služby podporujúce HTTP volania. Využitie spočíva v stiahnutí snímkov z webu Českého meteorologického ústavu.

Kapitola 5

Vyhodnotenie úspešnosti

Na vyhodnotenie úspešnosti bola použitá metóda CSI - Critical Success Index [3], niekedy nazývaná aj TS - Thread Score, po odporučení od doktora Petra Nováka (zamestnanca ČHMÚ).

Výpočet úspešnosti predpovede touto metódou pracuje s 3 hodnotami:

- úspešne predpovedaný jav - ***hits***
- neúspešne predpovedaný jav - ***false alarms***
- neúspešne nepredpovedaný jav - ***misses***

A je daný vzťahom:

$$CSI = TS = \frac{hits}{hits + false\ alarms + misses}$$

Výsledná hodnota sa nachádza v rozmedzí 0-1, pričom 1 znamená perfektnú predpoveď. Metóda CSI sa využíva v hodnotení meteorologických výstupov celkom často, zahŕňa aj nesprávne predpovedané javy a tak isto aj nepredpovedané javy. Preto je vyváženejšou hodnotou ako FAR alebo POD. Metóda však poskytuje slabšie výsledky pri vzácnych javoch. V aplikácii je to napríklad stav, kde dochádza k pohybu relatívne malých zhlukov zrážok.

Aplikácia má automatické vyhodnocovanie takýmto spôsobom. Vždy aplikuje túto metódu na vygenerovaný snímok a reálny, pričom snímky sú porovnané pixel po pixeli. Aktuálne percentá je možné vidieť priamo v aplikácii a k dátumu 8.5.2014 sú čísla nasledovné:

na 15 minút	na 30 minút	na 45 minút	na 60 minút
56%	42%	36%	32%

5.1 Zhodnotenie

Aplikácia funguje vo webovom prehliadači, čím je ideálna jej prenositelnosť, užívateľ je uchránený pred akoukolvek inštaláciou. Zvolené programovacie prostredie dobre korešponduje s navrhnutou štruktúrou aplikácie. V tomto prostredí sa pracovalo dobre a za najvačšiu zbraň tohto prostredia by som považovať ideálny spôsob komunikácie medzi jednotlivými

komponentami. Komunikácia na úrovni kódu medzi back-endom a databázou, alebo back-endom a front-endom bola velmi jednoduchá a spočívala v použití jedného programovacieho jazyka. Prepojenosť medzi vývojárskymi nástrojmi bola taktiež velmi dobrá. Git a Heroku sú výborne prepojené a aplikácia je na serveri nahratá v priebehu jednej minúty za použitia 3 príkazov v konzole.

Samotný algoritmus pracuje s obrázkami, to znamená že využíva velmi rozsiahle viacrozmerné polia a samotný výpočet je skôr náročný ako nenáročný. Preto masívne rozširovanie by aj predlžovalo tento výpočet. Aplikácia počíta rýchlo v danom stave tak ako je teraz, výpočet všetkcýh snímkov je ukončený do 10 sekúnd, pričom najviac času zaberá dekódovanie a zakódovanie obrázku.

5.2 Možné rozšírenia

Aplikácia vypočítava predpoveď zrážok, čo poskytuje množstvo rôznych ideí na rozširovanie.

Zadanie polohy

Užívateľ by si zadal polohu do mapy a tento údaj by sa predal serveru, ktorý by naspäť predal údaje o blížiacich sa zrážkach do zadaného miesta. Z postupu zrážok by sa dala vypočítať rýchlosť postupu zrážkových polí, to jest aj čas kedy by na dané miesto dorazili. Aplikácia by mohla informovať o presnom čase dorazenia na miesto, intenzite zrážok - čo sú údaje, ktoré by sa dali vyčítať z aplikácie tak ako vyzerá teraz. Plátno, v ktorom je animácia zrážok momentálne by bolo treba pravdepodobne prerobiť na Canvas.

Warning pred búrkami

ČHMÚ tak ako radarové snímky poskytuje aj snímky detekcie bleskov. Tieto dva zdroje by sa mohli dať dokopy a v animácii by sa ukazovali naraz aj zrážky aj identifikácia bleskov. Spolu s predchádzajúcim rozšírením by aplikácia vytvorila pekný nástroj na upozornenie pred búrkami. Podľa vzorcov by možno bolo možné vypočítať vietor a jeho nárazy, prípadne možnosť krupobytia, ktoré býva častým sprievodným javom najmä pri rozsiahlych supercelárnych búrkach.

Vylepšenie výpočtového algoritmu

Algoritmus nezohladňuje nijaké iné údaje pri výpočte ako radarové snímky. Preto ani nemá cenu meniť štruktúru zrážkových polí a táto časť bola z výpočtu uplne vynechaná. Zrážky sa totiž nesprávajú podla nejakej šablóny, štatistiky, zotrvačnosti (ich pohyb áno), ale pôsobí na ne kvantum fyzikálnych veličín. Keby sa tieto fyzikálne veličiny dali zohladniť vo výpočte, reálna úspešnosť a dôveryhodnosť by určite stúpla.

Kapitola 6

Záver

Vrámci tejto bakalárskej práce bola vytvorená webová aplikácia na predpoveď zrážok pre územie Českej republiky. K docieleniu predpovede vedie niekoľko krokov, využité sú netriviálne algoritmy pre zhlukovanie, výpočet a porovnávanie polygonov. Ucelený algoritmus vytvára predpoveď v podobe štyroch snímkov. Úspešnosť tohto algoritmu je vyhodnotená spôsobom CSI, ktorý sa bežne používa pri vyhodnocovaní numerických predpovedných modelov a percentuálne sa pohybuje v rozmedzí od 55 do 30%. Aplikácia beží na testovacom serveri Heroku, webový prehliadač užívateľa nie je zaťažený žiadnymi výpočtami, predpoveď sa generuje na serverovej časti.

Odrazy z meteorologického radaru sú použité so súhlasom Českého hydrometeorologického ústavu. Oddobrené bolo použitie týchto dát pre využitie v bakalárkej práci, preto *zatiaľ* nie je možné využívanie výslednej aplikácie komerčne.

Aplikácia poskytuje možnosti rozšírenia rôznymi smermi, čo do vylepšovania výpočtu, tak aj v interakcii s používateľmi. Dali by sa sem zaradiť komunikácia s GPS zariadením (ak by bolo dostupné) a následné upozorňovanie pred blížiacimi sa zrážkami smerujúcimi priamo na užívateľa, či skombinovanie radarových snímkov spolu výstupmi detektorov bleskov a vydávanie výstrah pred blížiacimi sa búrkami.

Literatura

[1] Amazon Simple Storage Service. http://aws.amazon.com/s3/, [cit. 2014-5-15].

[2] Android Developers. https://developer.android.com/index.html, [cit. 2014-5-15].

[3] Critical Success Index (CSI) or Threat Score (TS), and Equitable Threat Score (ETS). http://www.eumetcal.org/resources/ukmeteocal/temp/msgcal/www/english/msg/ver_categ_forec/uos2/uos2_ko4.htm, [cit. 2014-5-15].

[4] DBSCAN density data. http://en.wikipedia.org/wiki/File:DBSCAN-density-data.svg, [cit. 2014-5-15].

[5] Český hydrometeorologický ústav. http://www.chmi.cz/portal/dt?portal_lang=cs&menu=JSPTabContainer/P1_0_Home, [cit. 2014-5-15].

[6] Git. https://github.com/, [cit. 2014-5-15].

[7] Heroku. https://www.heroku.com/, [cit. 2014-5-15].

[8] MongoDB. http://www.mongodb.org/, [cit. 2014-5-15].

[9] Node.js. http://nodejs.org/, [cit. 2014-5-15].

[10] SAT (Separating Axis Theorem). http://www.codezealot.org/archives/55, [cit. 2014-5-15].

[11] Uptime Robot. https://uptimerobot.com/index.php#mainDashboard, [cit. 2014-5-15].

[12] w3schools. http://www.w3schools.com/, [cit. 2014-5-15].

[13] Kolektív autorov: *Geometria v príkladoch*. Didaktis, 2003, iSBN 80-89160-00-X.

[14] Kráčmar, R.: Meteorologické radiolokátory. http://portal.chmi.cz/files/portal/docs/meteo/rad/info_radar/, [cit. 2014-5-15].

[15] Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars: *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd rev. ed. 2008. 386 s., iSBN 978-3-540-77973-5.

[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. `http://www.shmu.sk/File/sms/ondras_nowcasting.pdf`, [cit. 2014-5-15].

[17] Ondráš, M.: Nowcasting na SHMÚ. `http://www.shmu.sk/File/sms/ondras_nowcasting.pdf`, [cit. 2014-5-15].

[18] Sobíšek, Bořivoj: *Meteorologický slovník výkladový terminologický: s cizojazyčnými názvy hesel ve slovenštině, angličtině, němčině, francouzštině a ruštině.* 1. vyd. Praha: Academia, 1993, 594 s. ISBN 80-85368-45-5.

[19] Souza, C.: K-Means Clustering. `http://crsouza.blogspot.cz/2010/10/k-means-clustering.html`, [cit. 2014-5-15].

[20] Zendulka, J. a. k.: *Získávání znalostí z databází.* FIT VUT v Brně, 2009, 160 s.