UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA

DIPLOMOVÁ PRÁCE

Využití analýzy obrazu k předpovědi počasí

Katedra matematické analýzy a aplikací matematiky
Vedoucí bakalářské práce: **Mgr. Ondřej Vencálek Ph.D.**
Vypracoval(a): **Bc. Jiří Štantejský**
Studijní program: N1103 – Aplikovaná matematika
Studijní obor Aplikace matematiky v ekonomii
Forma studia: prezenční
Rok odevzdání: 2017

BIBLIOGRAFICKÁ IDENTIFIKACE

Autor: Bc. Jiří Štantejský

Název práce: Využití analýzy obrazu k předpovědi počasí

Typ práce: Diplomová práce

Pracoviště: Katedra matematické analýzy a aplikací matematiky

Vedoucí práce: Mgr. Ondřej Vencálek Ph.D.

Rok obhajoby práce: 2017

Abstrakt: Tématem této diplomové práce je využití analýzy obrazu k předpovědi počasí. Práce je rozdělena do deseti na sebe navazujících kapitol. První 2 kapitoly se zabývají získáváním srážkových dat a jejich následným importováním do programu Wolfram *Mathematica*. Kapitoly 3 a 4 se věnují importovaným datům, konkrétně chybám v měření a míře zastoupení jednotlivých barev v každém pixelu. Kapitola 5 obsahuje způsob transformace získaných dat na nová, která jsou vhodnější pro další zpracování. Zbývajících 5 kapitol se zabývá následným zpracováním transformovaných dat a popisuje možnosti jejich užití pro predikci budoucích hodnot.

Klíčová slova: Analýza obrazu, počasí, těžiště, morfologická analýza, Mathematica

Počet stran: 53

Počet příloh: 1(cd)

Jazyk: český

BIBLIOGRAPHICAL IDENTIFICATION

Author: Bc. Jiří Štantejský

Title: Image analysis for weather forecast

Type of thesis: Master's thesis

Department: Department of Mathematical Analysis and Applications of Mathematics

Supervisor: Mgr. Ondřej Vencálek Ph.D.

The year of presentation: 2017

Abstract: The theme of this thesis is the usage of image analysis for weather forecasting. The thesis is divided into ten consecutive chapters. The first two chapters contain acquiring of precipitation data and their subsequent import into Wolfram *Mathematica*. In chapters 3 and 4 We work with the imported data, specifically with observation errors and the degree of representation of individual colors in each pixel. Chapter 5 provides a way to transfrom the obtained data into new ones that are more suitable for further processing. The remaining 5 chapters deal with the subsequent processing of the transformed data and describe the options of their usage for prediction of future values.

Key words: Image analysis, weather, Center of gravity, Morphology analysis, Mathematica

Number of pages: 53

Number of appendices: 1(cd)

Language: Czech

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně pod vedením pana Mgr. Ondřeje Vencálka Ph.D. a všechny použité zdroje jsem uvedl v seznamu literatury.

V Olomouci dne 27.4.2017 .....

podpis

Obsah

Poděkování

Mé poděkování patří Mgr. Ondřeji Vencálkovi Ph.D. za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování diplomové práce věnoval.

Úvod

Počasí ovlivňuje většinu živých organismů na Zemi, a proto se jej snaží lidé od nepaměti předpovídat. Předpovědi počasí se v průběhu času vyvíjí, od jednoduchých pranostik až například po numerický model Aladin, který vyvinul a stále vylepšuje Český hydrometeorologický ústav.

Základem numerického modelu Aladin jsou radarová data ze dvou českých radarů, které měří aktuální srážkovou oblačnost nad územím České republiky, a právě tato data jsou základem i této diplomové práce. Cílem této diplomové práce je na základě získaných radarových měření predikovat budoucí vývoj těchto srážek.

V prvních čtyřech kapitolách se postupně zabýváme informacemi, které nám přibližují zkoumaná data. Můžeme zde nalét informace, jak jsou data měřena, jakým způsobem je můžeme importovat do programu *Mathematica* a jak jednotlivé obrázky vypadají, pokud je převedeme na datovou matici. V páté kapitole si ukážeme možnosti transformace získaných dat na nová, která budeme následně zkoumat. Ve zbývajících pěti kapitolách nalezneme možnosti využití dat pro predikci budoucích srážek. Prezentovány jsou vhodnější metody predikce (například využití mofrologické analýzy), tak i ty méně vhodné (například metoda těžiště).

Veškeré výpočty této práce jsou realizovány v programu Wolfram *Mathematica.* Jedná se o velice komplexní program, který nabízí velké množství funkci. Nedílnou součástí každé verze programu je velmi propracovaná nápověda, která ke každé funkci uvádí nejenom její zápis a popis, ale vždy obsahuje i mnoho názorných příkladů. Celý systém propracované nápovědy je doplněn oficiálním blogem (blog.wolfram.com), na kterém můžeme nalézt stovky článků od uživatelů tohoto programu, kde prezentují možnosti jeho využití. Program *Mathematica* může sloužit i jako textový editor, který má v sobě zakomponovány například i funkce pro generování obsahu a seznamu literatury. Jelikož ale program plně nepodporuje českou abecedu, jsou úvodní a konečné strany tohoto dokumentu vysázeny pomocí LaTeXu.

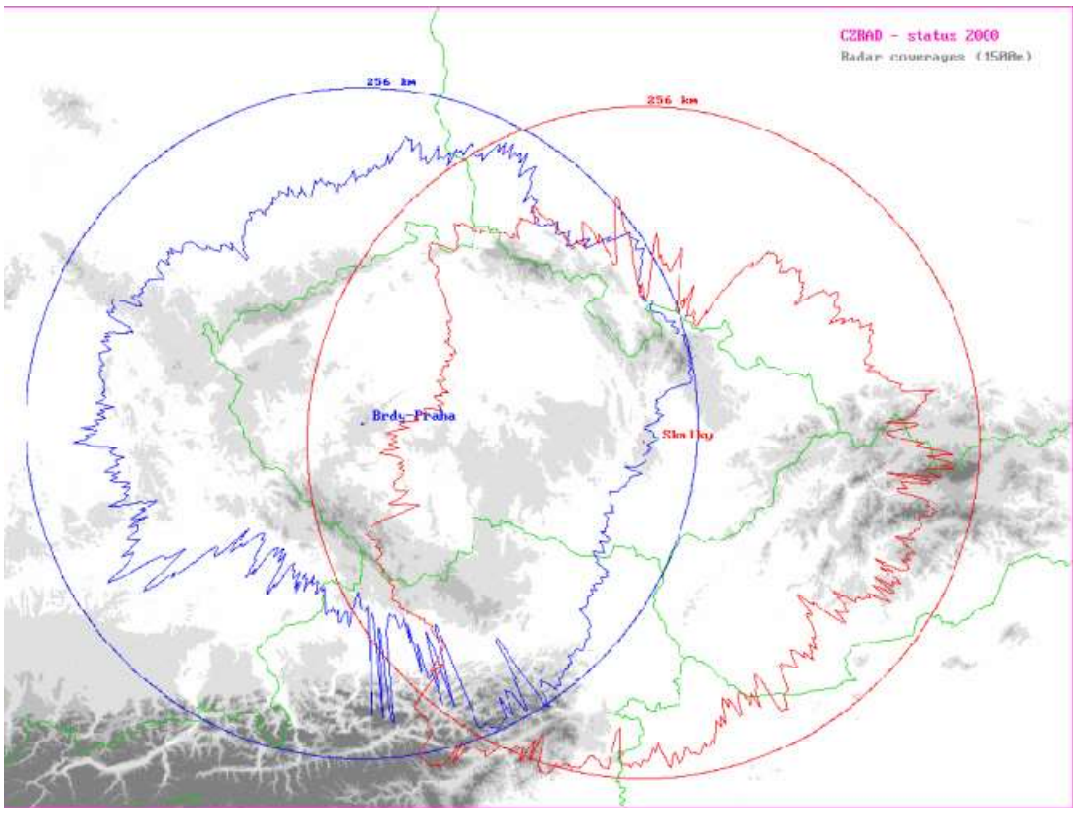
1 Zdroje dat

V České republice se pro detekci výrazné srážkové oblačnosti používají dva identické meteorologické radiolokátory. K jejich obnově došlo naposledy v roce 2015, na základě veřejné zakázky v částce 58 132 727 Kč, kterou realizovala společnost OMNIPOL a.s. Radary poskytují třírozměrnou informaci o srážkových jevech v atmosféře ve vysokém prostorovém (1x1 km) i časovém (5 minut) rozlišení. Data ze dvou českých radarů sítě CZRAD (radar Brdy v Čechách a radar Skalky na Moravě) pokrývají celou Českou republiku a její blízké okolí. [5,6]

Radar Skalky u Protivanova je umístěn v oblasti Střední morava na souřadnicích 49,501° severní šířky a 13,790° východní délky. Nachází se v nadmořské výšce 730 metrů nad mořem a samotná anténa je umístěna o 37 metrů výše. [5,6]

Radar Brdy - Praha je umístěn v oblasti Střední čechy na souřadnicích 49,658° severní šířky a 13.818° východní délky. Nachází se v nadmořské výšce 860 metrů nad mořem a samotná anténa je umístěna o 56 metrů výše. [5,6]

Radary jsou provozovány od roku 1995 (Skalky u Protivanova) a 1999 (Brdy - Praha) a jejich radarová část prošla modernizací v roce 2015, jak je již zmíněno výše. Radarová část je na obou radarech identická a tvoří jí radar Vaisala WRM200. Primárním cílem obnovy radarů bylo zajištění dostatečně spolehlivých radarových měření i v budoucích letech, neboť do té doby užívané radary byly již na konci své životnosti, což se projevovalo převážně jejich sníženou spolehlivostí. Nově instalované radary umožňují navíc provádět polarimetrická měření, která zvyšují kvalitu dat, mj. i zkvalitnění standardních radarových dat radarové odrazivosti a z nich počítaných odhadů srážek. [5,6]



Obrázek 1: Maximální dosahy meteorologických radarů ČHMÚ (kruhy) a dosahy pro určování intenzit srážek (do výšky 1500 m nad terénem) dle doporučení projektu COST 73. [1]

Na obrázku číslo 1 můžeme sledovat maximální dosahy meteorologických radarů a dosahy pro určování internzit srážek. Obrázek je konstruován pro předchůdce současných radarů Vaisala, které mají maximální dosah 260 km, tudíž i dosah pro určování srážek je větší. Jelikož se ale maximální dosah liší pouze o 4 km a dosah pro určování srážek je určen především terénem na povrchu, který je v krátkobém měřítku neměnný, můžeme tento obrázek, pro ilustraci, považovat za dostatečný.

Obrázek 2: Anténa radaru Vaisala WRM200 umístěná na radarové základně Skalky u Protivanova. [7]

Obrázek 3: Radarová základna Skalky u Protivanova. [11]

Obrázek 4: Radarová základna Brdy - Praha. [12]

2 Načítání dat

Data z radarové sítě používá veliké množství zpravodajských serverů. Již při zadání diplomové práce jsme určili, že budeme požívat data ze serveru "http://pocasi.idnes.cz/", konkrétně srážková data ze serveru "http://pocasi.idnes.cz/?t=img_s", která automaticky naimportujeme přímo do programu Wolfram Mathematica®.

Pro importování dat je nejvhodnější použít funkci "Import". Jedná se o velmi všestranou fnkci, která dokáže zpracovat velké množství formátů, z nichž každý může obsahovat veliké množství prvků. Současně nám tato funkce umožňuje importovat pouze vybraný typ prvků, např. pouze obrázky.

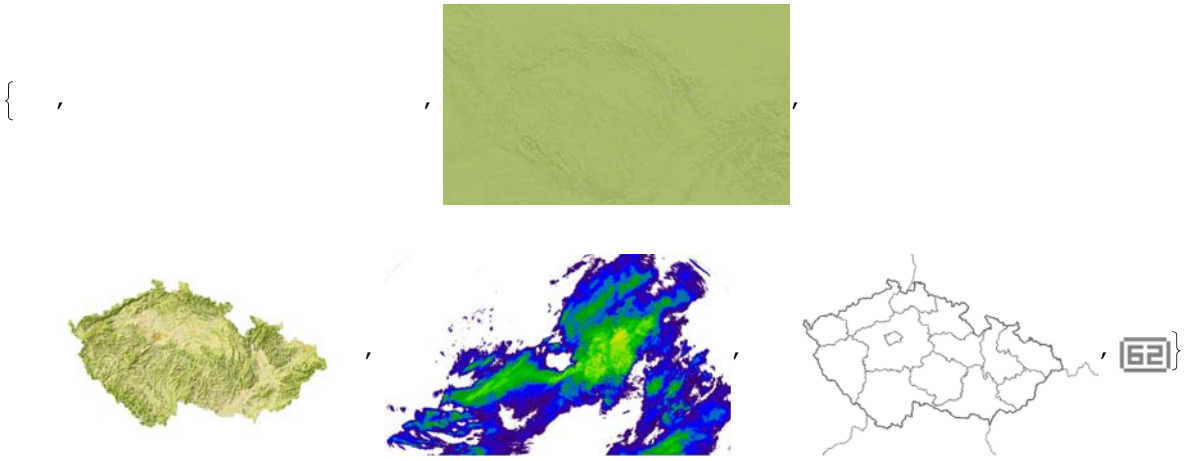
Import ["*file"*] - importuje všechna data ze souboru

Import ["*file", elements*] - importuje konkrétní zvolený typ prvků ze souboru

Import ["http://*url", ...*] a Import ["ftp://*url", ...*] - importuje ze zvoleného typu URL adresy

V této diplomové práci používám funkci Import dvěmi způsoby:

```
podklad = Import["http://pocasi.idnes.cz/?t=img_s", "Images"]
```



Tímto způsobem importujeme obrázky, které server idnes.cz používá jako neměnné mapové podklady, konkértně mapu krajů a reliéf České republiky. Tyto obrázky budeme následně používat pro grafický výstup. Jelikož ale stránka používá javascript, není možné tímto způsobem získat konkrétní datové obrázky se srážkami, ale jednotlivé obrázky je nutné získat přímo ze zdrojového kódu stránky.

```
d = Import["http://pocasi.idnes.cz/?t=img_s", "Source"];
```

Tímto způsobem do proměnné d naimportujeme celý zdrojový kód stránky. Ze zdrojového kódu můžeme "extrahovat" URL adresy, ze kterých budeme následně importovat radarové snímky. Proto nás zajímá tato část:

<a onclick="return !Pocasi.animuj('meteoImage', ['//1gr.cz/pocasi/17/04/radar/5283368_pacz23.z_max3d.20170410.1630.0_n.png','//1gr.cz/pocasi/17/04/radar/5283398_pacz23.z_max3d.20170410.1645.0_n.png','//1gr.cz/pocasi/17/04/radar/5283432_pacz23.z_max3d.20170410.1700.0_n.png','//1gr.cz/pocasi/17/04/radar/5283468_pacz23.z_max3d.20170410.1715.0_n.png','//1gr.cz/pocasi/17/04/radar/5283503_pacz23.z_max3d.20170410.1730.0_n.png','//1gr.cz/pocasi/17/04/radar/5283533_pacz23.z_max3d.20170410.1745.0_n.png','//1gr.cz/pocasi/17/04/radar/5283567_pacz23.z_max3d

.20170410.1800.0_n.png','//1gr.cz/pocasi/17/04/radar/5283597_pacz23.z_max3d.20170410.1815.0_n.png','//1gr.cz/pocasi/17/04/radar/5283638_pacz23.z_max3d.20170410.1830.0_n.png','//1gr.cz/pocasi/17/04/radar/5283668_pacz23.z_max3d.20170410.1845.0_n.png','//1gr.cz/pocasi/17/04/radar/5283698_pacz23.z_max3d.20170410.1900.0_n.png','//1gr.cz/pocasi/17/04/radar/5283734_pacz23.z_max3d.20170410.1915.0_n.png','//1gr.cz/pocasi/17/04/radar/5283769_pacz23.z_max3d.20170410.1930.0_n.png','//1gr.cz/pocasi/17/04/radar/5283800_pacz23.z_max3d.20170410.1945.0_n.png','//1gr.cz/pocasi/17/04/radar/5283836_pacz23.z_max3d.20170410.2000.0_n.png','//1gr.cz/pocasi/17/04/radar/5283866_pacz23.z_max3d.20170410.2015.0_n.png'],'time_box',
['10.4.
18:30
1 / 16 ','10.4.
18:45
2 / 16 ','10.4.
19:00
3 / 16 ','10.4.
19:15
4 / 16 ','10.4.
19:30
5 / 16 ','10.4.
19:45
6 / 16 ','10.4.
20:00
7 / 16 ','10.4.
20:15
8 / 16 ','10.4.
20:30
9 / 16 ','10.4.
20:45
10 / 16 ','10.4.
21:00
11 / 16 ','10.4.
21:15
12 / 16 ','10.4.
21:30
13 / 16 ','10.4.
21:45
14 / 16 ','10.4.
22:00
15 / 16 ','10.4.
22:15
16 / 16 '], this, 'idbox',
['1041830','1041845','1041900','1041915','1041930','1041945','1042000','1042015','1042030','1042045','1042100','1042115','1042130','1042145','1042200','1042215']);"
href="#" id="a-play">

Použitím funkce StringSplit následně oddělíme část zdrojového kódu, která se nachází před a za částí s odkazy, kterou uložíme do proměnné w.

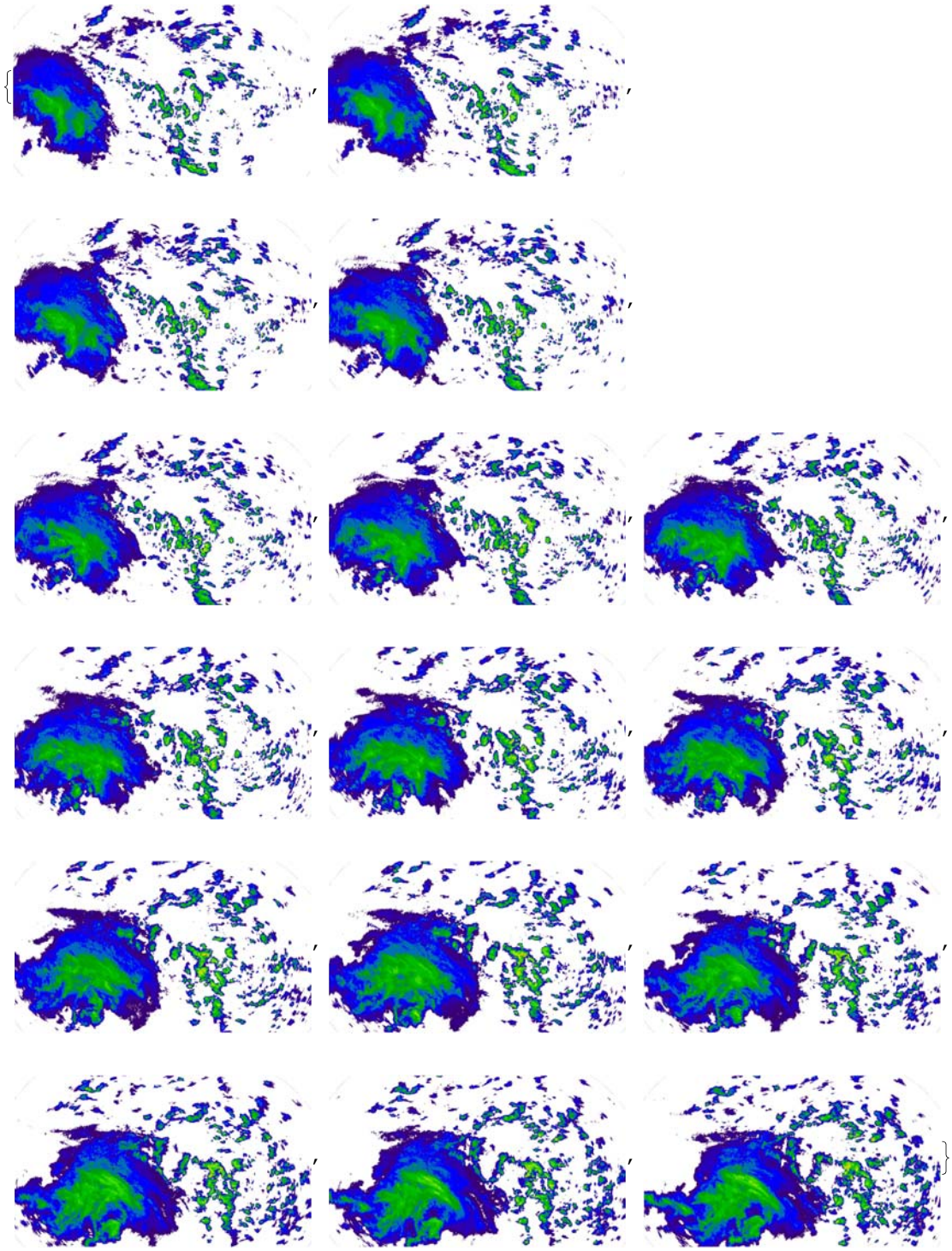
```
v = StringSplit[d, "!Pocasi.animuj"];
vv = StringSplit[v[[2]], "time_box"] ;
w = vv[[1]];
```

Získané adresy bohužel neobsahují, pro Mathematicu důležitou, předponu http://, a proto jí doplníme funkcí StringReplace. Pomocí této funkce je možné nahradit část (části) textu jiným, dle zvolených pravidel. Současně můžeme určit, zda tuto náhradu chceme provést v celém dokumentu nebo jen v prvních několika případech.

```
ww = StringReplace[w, {"//" -> "http://", "'" → ""}];
www = StringSplit[ww, {"[", ",", "]"}];
adresy = www[[3 ;; 18]]
```

```
{http://1gr.cz/pocasi/17/04/radar/5305553_pacz23.z_max3d.20170417.0945.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305587_pacz23.z_max3d.20170417.1000.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305617_pacz23.z_max3d.20170417.1015.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305653_pacz23.z_max3d.20170417.1030.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305686_pacz23.z_max3d.20170417.1045.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305727_pacz23.z_max3d.20170417.1100.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305757_pacz23.z_max3d.20170417.1115.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305792_pacz23.z_max3d.20170417.1130.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305822_pacz23.z_max3d.20170417.1145.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305857_pacz23.z_max3d.20170417.1200.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305887_pacz23.z_max3d.20170417.1215.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305929_pacz23.z_max3d.20170417.1230.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305959_pacz23.z_max3d.20170417.1245.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5305994_pacz23.z_max3d.20170417.1300.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5306025_pacz23.z_max3d.20170417.1315.0_n.png,
 http://1gr.cz/pocasi/17/04/radar/5306060_pacz23.z_max3d.20170417.1330.0_n.png}
```

```
zaznam = Table[Import[adresy[[c]]], {c, 1, Length[adresy]}]
```



Touto úpravou získáme seznam odkazů na 16 nejaktuálnějších obrázků, které popisují intenzitu srážek na území České republiky a v přilehlém okolí. Pomocí funkce Table následně tato data můžeme zobrazit a dále s nimi pracovat.

Pro lepší představu o pohybu srážek získané obrázky doplníme o mapu krajů České republiky a o její reliéf použitím funkce ImageCompose. Následným použitím funkce ListAnimate získaná data "naanimujeme". Funkce ImageCompose umožňuje sloučit, resp. překrýt jeden obrázek druhým. Navíc můžeme určit i intenzitu překrývajícího obrázku, případně jeho polohu. Možnosti zápisu a

použití jsou následující:

ImageCompose [*image*, *overlay*] - výsledkem je plné překrytí obrázku (image) obrázkem (overlay)

ImageCompose [*image*, {*overlay,* α}] - výsledkem je překrytí obrázku (image) obrázkem (overlay) v intenzitě $\alpha \in \{0,1\}$

ImageCompose [*image*, *overlay, pos*] - výsledkem je překrytí obrázku (image) obrázkem (overlay), kdy střed obrázku (overlay) je umístěn na pozici pos

ImageCompose [*image*, *overlay, pos, opos*] - výsledkem je překrytí obrázku (image) obrázkem (overlay), kdy bod (opos) obrázku (overlay) je umístěn na pozici pos obrázku (image)

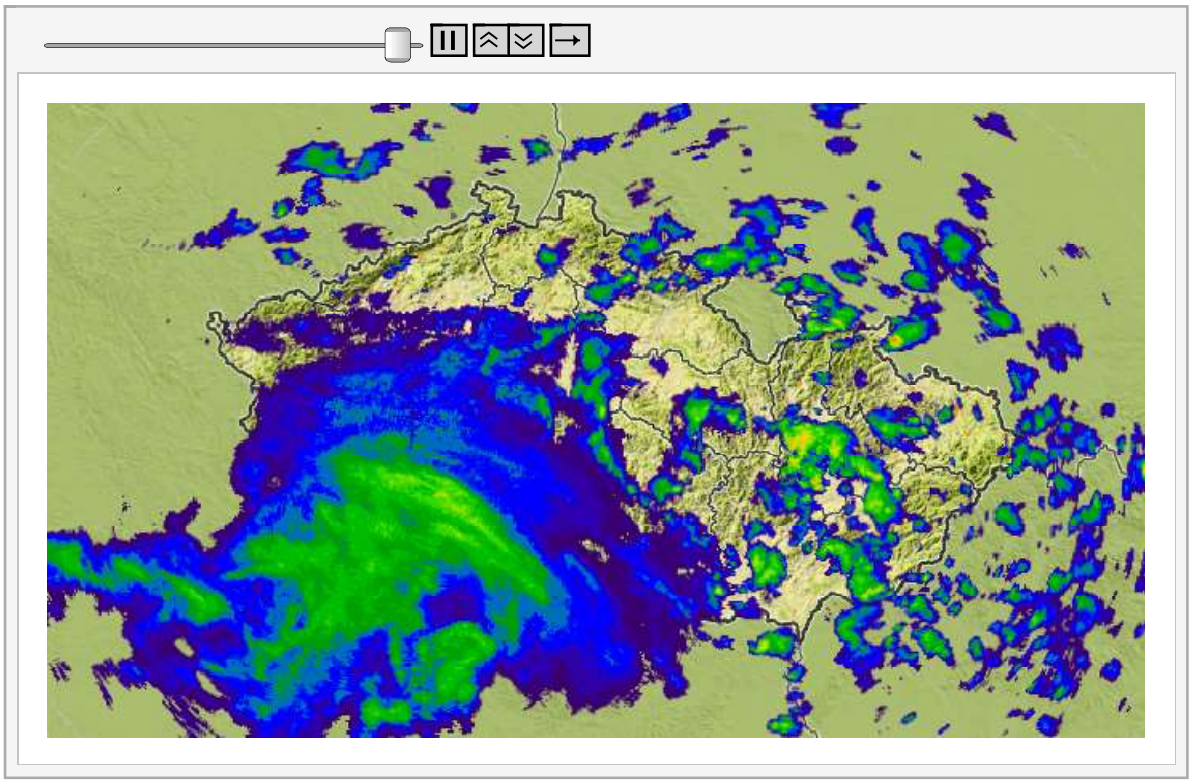
Funkce ListAnimate umožní provést animaci obrázků pro lepší představu o pohybu a změně zkoumaných srážek. Animaci srážek využívá většina serverů, které měřené hodnoty prezentuje, včetně poskytovatele dat Českého hydrometeorologického ústavu i serveru pocasi.idnes.cz, ze kterého data importujeme. Základní možnosti zápisu jsou následující:

ListAnimate [{$expr_1$, $expr_2$, ...}] - generuje animaci z po sobě jdoucích obrázků $expr_i$

ListAnimate [*list, fps*] - pomocí *fps* je možné nastavit počet obrázků za sekundu (frames per second)

Již při zadání funkce ListAnimate je možné volit další možnosti chování animace, jako například směr, počet opakování apod. případně je měnit přímo v animačním okně. Výslednou animaci v našem případě vytvoříme použitím následujících příkazů:

```
podklady = ImageCompose[ImageCompose[podklad[[3]], podklad[[4]]], podklad[[6]]];
dataproanimaci = Table[ImageCompose[podklady, zaznam[[i]]], {i, 1, Length[zaznam]}];
ListAnimate[dataproanimaci]
```



```
Export["animacesrazek.swf", %];
```

Jelikož zde prezentovanou animaci bohužel není možné vytisknout, její záloha s názvem *animacesrazek.swf* bude přiložena jako příloha na CD.

Další velmi často užívané funkce při analýze obrazu jsou funkce Image a ImageData. Pomocí funkce Image můžeme data přetransformovat do podoby obrázku, stejně tak pomocí funkce ImageData obrázek přetransformovat na datovou množinu. Zmíněné funce mají následující strukturu zápisu:

Image [*data*] - vytvoří obraz z pixelů, které jsou definované polem *data*

Image [*graphics*] - vytvoří obraz z jiných grafických objektů

Image [*obj, options*] - vytvoří obraz za použití specifických možností, resp. požadavků

ImageData [*image*] - vytvoří pole hodnot pro příslušné pixely obrazu

ImageData [*image, "type"*] - vytvoří pole hodnot pro příslušné pixely obrazu, které přetransformuje na požadovaný typ hodnot

```
data = Table[ImageData[zaznam[[i]]], {i, 1, Length[zaznam]}];
Dimensions[data]
```

```
{16, 375, 648, 4}
```

Použitím funkce ImageData na námi získané obrázky získáme čtyřrozměrnou matici, která obsahuje více jak 15,5 miliónu hodnot, které budeme následně zpracovávat. První hodnota nám udává počet získaných obrázků, tedy 16 nejaktuálnějších. Následující hodnoty udávají velikost každého

získaného obrázku, tedy 375 x 648 pixelů. Jelikož radary Vaisala provádí měření v rozlišení 1x1 km, výsledné rozměry v pixelech odpovídají i rozměrům v km, tedy každý obrázek pokrývá oblast velikosti 375x648=243 tis. km². Poslední údaj udává počet hodnot v každém pixelu, tedy 4.

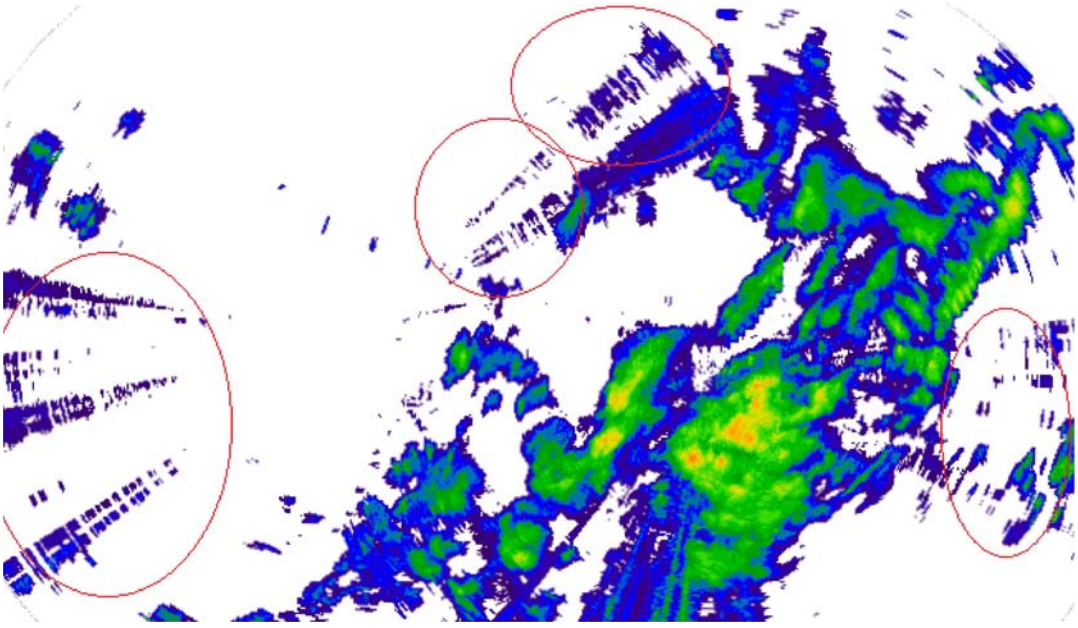
```
data[[1, 150, 400]]
```

```
{0.211765, 0., 0.486275, 1.}
```

Jednotlivé pixely obsahují hodnoty udávající míru zastoupení čtyř základních složek barevného schématu RGBA, tedy první položka míru zastoupení červené barvy, druhá položka míru zastoupení zelené barvy, třetí položka míru zastoupení modré barvy a alfa kanál A s informací o průhlednosti konkrétního pixelu. Míry zastoupení jednotlivých barev náleží do intervalu <0,1>. Alfa kanál je v našem případě vždy roven jedné a pro následující zpracování nevýznamný.

3 Chyby v měření

Ani v případě tvorby srážkových map nejsou data vždy přesná a mnohdy obsahují chyby v měření. Nejčastěji pozorovanou chybou jsou pravidelné, ostře ohraničené obrazce tvaru trojúhelníku (šipky), které směřují do jednoho ze dvou středů na mapě. V těchto středech jsou umístěny radary Českého hydrometeorologického ústavu.



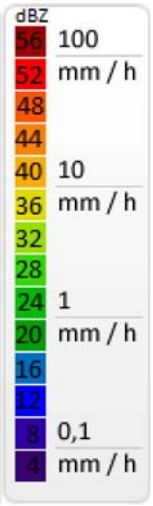
Obrázek 5: Znázorněné chyby měření.

Tento druh chyby je způsoben provozovateli bezdrátových technologií RLAN na frekvenci 5GHz (provozovatelé některých WiFi sítí). Jelikož je toto rušení velmi nežádoucí, spolupracuje ČHMÚ s ČTÚ na snížení množství rušitelů. Konkrétní seznam těchto rušitelů uveřejňuje ČHMÚ pravidelně na svých stránkách. [9]

Většinu těchto chyb následně odstraníme pomocí škálování dat, kterému se věnuje následující kapitola.

4 Tvorba škály

Intenzita srážek je měřena pomocí decibelů Z, zkráceně dBZ, což jsou jednotky radarové odrazivosti. V těchto jednotkách je následně definována barevná škála, dle které je možné opticky určit intenzitu srážek, resp. srážkové úhrny.



Obrázek 6: Škála udávající intenzitu srážek v mm/h a dBZ.[5, upraveno]

Intenzita	Barva [dBZ]	Srážkové úhrny	Projev
Slabé	fialová, modrá (4 – 16 dbz)	pod 0.5 mm/h	Mrholení, kapání
Středně silné	zelená (20 – 32 dbz)	kolem 4 mm/h	Déšť, je potřeba deštník, silnice jsou lesklé
Silné	žlutá, oranžová (36 – 44 dbz)	kolem 10 mm/h	Silný déšť
Velmi silné	červená (48 – 56 dbz)	kolem 70 mm/h	Přívalový déšť, bouřky, hrozí škody na majetku
Extrémní	bílá (60 dbz)	nad 100 mm/h	Většinou krupobití

Tabulka 1: Určení intenzity srážek. [8]

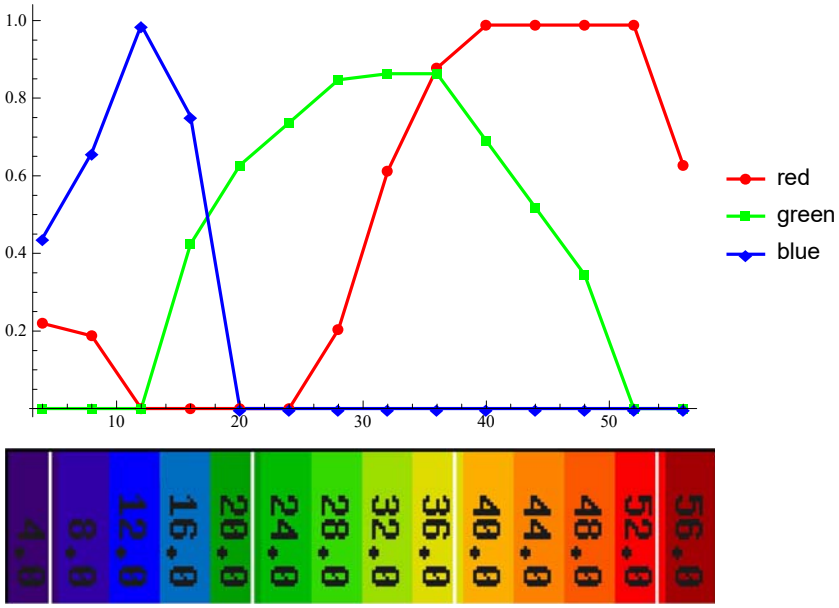
Abychom dokázali určit intenzitu srážek v importovaném obrázku, je nutné zjistit, v jakých úrovních a poměrech se vyskytují barvy RGB pro jednotlivé intenzity. V případě použití dané škály získáme následující hodnoty:

Odrazivost/Barva	Red	Green	Blue
4	0.22	0	0.439
8	0.188	0	0.659
12	0	0	0.988
16	0	0.424	0.753
20	0	0.627	0
24	0	0.737	0
28	0.204	0.847	0
32	0.612	0.863	0
36	0.878	0.863	0
40	0.988	0.69	0
44	0.988	0.518	0
48	0.988	0.345	0
52	0.988	0	0
56	0.627	0	0

Tabulka 2: Zatoupení jednotlivých barev pro udané odrazivosti z barevné škály.

Pro lepší názornost vykreslíme jednotlivé hodnoty do grafu:

```
IntenzityB = {{0.22, 0., 0.439}, {0.188, 0., 0.659},
    {0., 0., 0.988}, {0., 0.424, 0.753}, {0., 0.627, 0.}, {0., 0.737, 0.},
    {0.204, 0.847, 0.}, {0.612, 0.863, 0.}, {0.878, 0.863, 0.}, {0.988, 0.69, 0.},
    {0.988, 0.518, 0.}, {0.988, 0.345, 0.}, {0.988, 0., 0.}, {0.627, 0., 0.}};
ListLinePlot[Transpose[IntenzityB], PlotLegends → {"red", "green", "blue"},
 PlotMarkers → Automatic, PlotStyle -> {Red, Green, Blue},
 DataRange → {4, 56}, PlotRange → All]
```



Obrázek 7: Graf zastoupení jednotlivých barev ve škále doplněný o škálu.

5 Transformace dat

Jak je již uvedeno výše, aktuálně pracujeme s více než 15,5 miliony hodnot. Toto množství dat je pro běžný počítač výpočetně náročné, a proto je vhodné jejich počet zredukovat. Stejně tak, při použití vhodných škál, můžeme docílit i dalších pozitivních jevů. Pro další výpočty jsem se rozhodl rozdělit data do pěti okruhů dle následujícího klíče.

Nová hodnota	Podmínky			
0	R < 0.1	G < 0.1	B < 0.1	jinak
0	B > 0			jinak
0.04	G > R			jinak
0.1	R > G	G >= 0.5		jinak
0.7	R >= 0.95	G < 0.5		jinak
1	(R < 0.95)			

Podmínky pro transformaci dat byly zvoleny s ohledem na významnost jednotlivých srážek. V prvním kroku zkoumáme drobné, spíše zaokrouhlovací, chyby, a proto v případě, že jsou všechny hodnoty menší jak 0.1, přiřazujeme jim hodnotu 0. Stejně tak za nevýznamné srážky považujeme srážky s hodnotou odrazivosti menší jak 20 dBZ. Jedná se o fialově a modře zbarvené srážky, které odpovídají mrholení či občasnému kapání. V této intenzitě je také zaznamenávána většina chyb (viz kapitola 4) a touto transformací způsobíme jejich eliminaci. Pro středně silné srážky, tj. srážky s hodnotou odrazivosti přibližně 20 - 36 dBZ, je přiřazována hodnota 0.04. Jedná se o srážky v odstinech zelené a jejich projevem je mírný déšť. Hodnoty 0.1 jsou přiřazovány silným srážkám, jejichž hodnoty odrazivosti jsou přibližně 36 - 44 dBZ. Srážky se projevují silným deštěm a na získaném obrázku mají žlutou až oranžovou barvu. Za velmi silné srážky považujeme červeně podbarvené srážky s hodnotou odrazivosti přibližně 44 - 54 dBZ. Jedná se o přípavlové srážky, často doprovázené bouřkami. Těmto srážkám přiřazujeme hodnotu 0.7. Nejintenzivnější, resp. extrémní srážky jsou barevně znázorněny tmavě červenou. Jejich hodnota odrazivosti je nad 54 dBZ a přiřazujeme jim hodnotu 1. Na mapách můžeme občas vidět i extrémní stupeň srážek, které jsou značeny bíle, tedy hodnoty RGB jsou rovny jedné. V tomto případě se jedná většinou o krupobití. Jelikož se ale jedná o velmi ojedinělý jev, který se navíc vyskytuje pouze lokálně a krátkodobě, budeme jej v dalších výpočtech zanedbávat. Hodnoty 0.04, 0.1 a 0.7 byly zvoleny záměrně, neboť se jedná o průměrný srážkový úhrn v mm za hodinu.

Pro transformaci čtveřice dat na jednu novou použijeme funkci Which, která postupně testuje podmínky. V případě, že je podmínka splněna, cyklus se ukončí. V případě, že jsou v jednom cyklu splněny 2 a více podmínek, přednost má ta dřívější. Abychom pokryli všechny hodnoty, použijeme tří funkcí For v rozsahu od jedné až po maximální dimenze dat, tj. 16, 375, 648. Funkce For a Which mají následující zápisy:

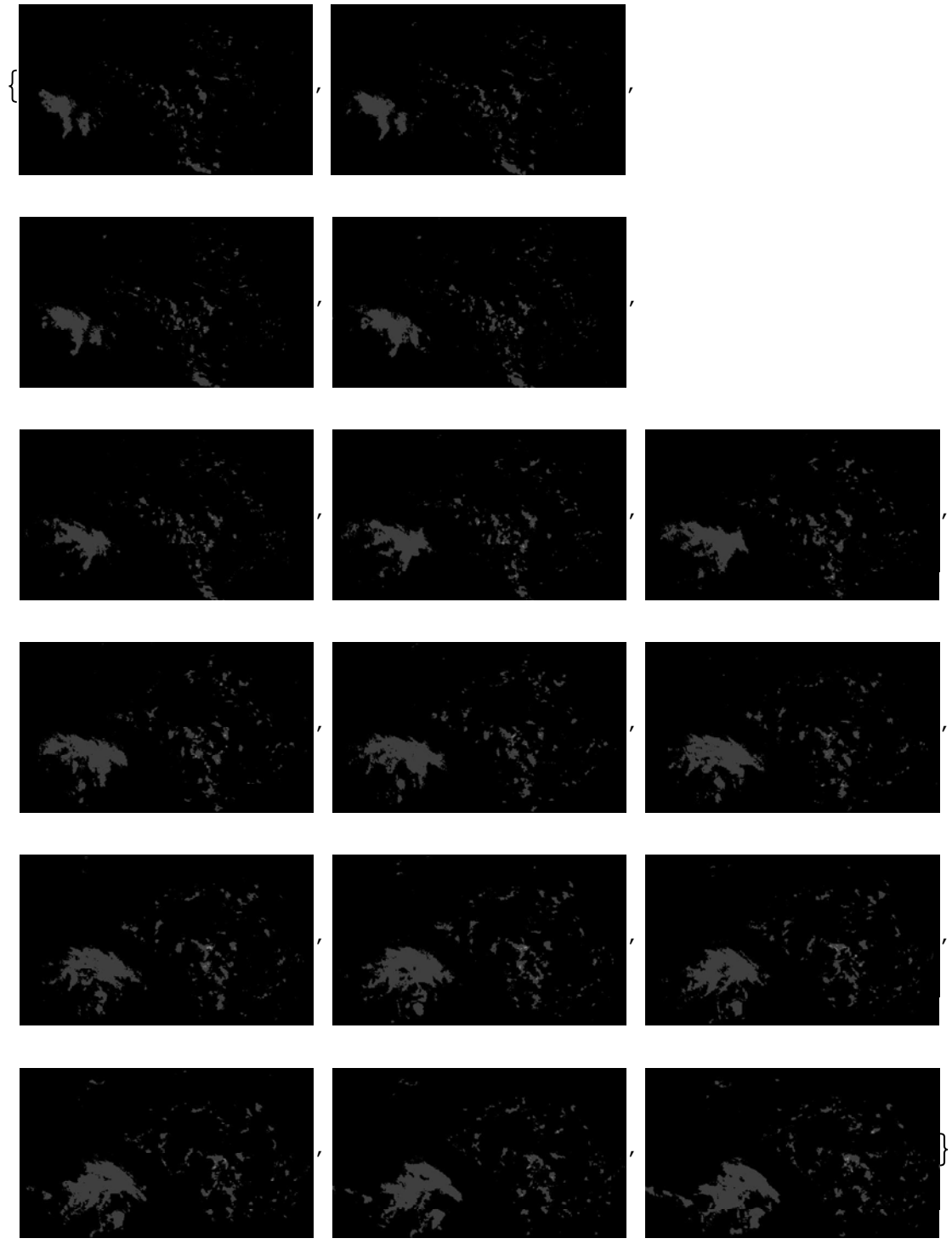
For [*start, test, incr, body*] - funkce For provede pokyn na pozici *start.* Následně vyhodnocuje příkazy na pozicích *incr* a *body*, dokud je plněna podmínka na pozici *test.* V případě, že podmínka na pozici *test,* je vyhodnocena negativně, cyklus se ukončí.

Which [$test_1$, $value_1$, $test_2$, $value_2$, ...] - tato funkce postupně vyhodnocuje podmínky na pozici $test_i$ a v případě splnění dané podmínky vrací hodnotu $value_i$.

Pro lepší grafickou prezentaci provedeme 2 transformace, jelikož hodnoty 0.04, které se po nule vyskytují nejčastněji, jsou téměř černé a grafická prezentace je nedostatečná. Proto provedeme i transformaci na hodnoty {0,0.25,0.5,0.75,1}. Výpočty ale budeme provádět s hodnotami transformovanými dle kritérií uvedených výše.

Importovaná data přetransformujeme následující skupinou příkazů:

```
data = Table[ImageData[zaznam[[c]]], {c, 1, Length[zaznam]}];
For[k = 1, k <= Dimensions[data][[1]], k++,
 For[i = 1, i <= Dimensions[data][[2]], i++,
  For[j = 1, j <= Dimensions[data][[3]], j++,
   Which[data[[k, i, j, 1]] < 0.1 &&
     data[[k, i, j, 2]] < 0.1 && data[[k, i, j, 3]] < 0.1, data[[k, i, j]] = 0,
    data[[k, i, j, 3]] > 0, data[[k, i, j]] = 0,
    data[[k, i, j, 2]] > data[[k, i, j, 1]], data[[k, i, j]] = 0.25,
    data[[k, i, j, 1]] >= data[[k, i, j, 2]] && data[[k, i, j, 2]] >= 0.5,
    data[[k, i, j]] = 0.5,
    data[[k, i, j, 1]] >= 0.95 && data[[k, i, j, 2]] < 0.5, data[[k, i, j]] = 0.75,
    data[[k, i, j, 1]] < 0.95, data[[k, i, j]] = 1]
  ]]]
dataobr = Table[Image[data[[c]]], {c, 1, Length[data]}]
```



```
Export["data170417_1.mat", data];
```

```
data1 = Table[ImageData[zaznam[[c]]], {c, 1, Length[zaznam]}];
For[k = 1, k <= Dimensions[data1][[1]], k++,
 For[i = 1, i <= Dimensions[data1][[2]], i++,
  For[j = 1, j <= Dimensions[data1][[3]], j++,
   Which[data1[[k, i, j, 1]] < 0.1 &&
     data1[[k, i, j, 2]] < 0.1 && data1[[k, i, j, 3]] < 0.1, data1[[k, i, j]] = 0,
    data1[[k, i, j, 3]] > 0, data1[[k, i, j]] = 0,
    data1[[k, i, j, 2]] > data1[[k, i, j, 1]], data1[[k, i, j]] = 0.04,
    data1[[k, i, j, 1]] >= data1[[k, i, j, 2]] && data1[[k, i, j, 2]] >= 0.5,
    data1[[k, i, j]] = 0.1,
    data1[[k, i, j, 1]] >= 0.95 && data1[[k, i, j, 2]] < 0.5, data1[[k, i, j]] = 0.7,
    data1[[k, i, j, 1]] < 0.95, data1[[k, i, j]] = 1]
  ]]]
Export["data170417_2.mat", data1];
```

Jelikož transformace těchto dat je výpočetně relativně náročná, obě skupiny jsou vyexportovány do souborů *data170417_1.mat* a *data170417_2.mat*, aby je bylo možné kdykoliv opět naimportovat bez nutnosti opětovného výpočtu. Tato data budou také součástí přiloženého CD. Pro importování dat opět slouží funkce Import, kterou je nutné doplnit o "cestu" k požadovaným datům.

```
data1704171 = Import[".../data170417_1.mat"];
data1704172 = Import[".../data170417_2.mat"];
```

Transformací dat získáme především zjednodušení jejich struktury, neboť 4 hodnoty jsou nahrazeny jednou. Současně získáme data s jejichž růstem roste i jejich intenzita a v neposlední řadě dojde k redukci většiny chyb v měření.

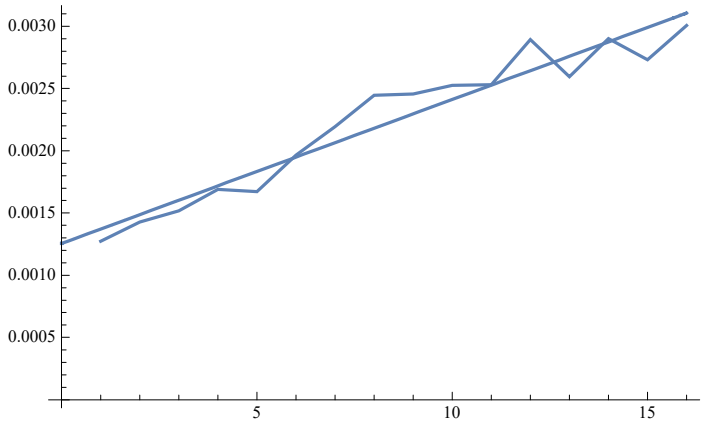
6 Množství srážek v čase

Pro budoucí odhad vývoje srážek je vhodné sledovat jejich množství v čase pro celou pozorovanou oblast. Pomocí transformovaných dat zjistíme, zda množství srážek v čase přibývá, ubývá či se výrazně nemění.

```
celksr = Table[0, Dimensions[data1704172][[1]]];
For[i = 1, i ≤ Dimensions[data1704172][[1]], i++,
celksr[[i]] = Total[data1704172[[i]], 2]]
celksra = celksr / (375 * 648)
model = LinearModelFit[celksra, x, x]
Show[ListLinePlot[celksra], Plot[model["BestFit"], {x, 0, 16}]]
```

```
{0.00127251, 0.00142708, 0.00151572, 0.00168971, 0.00167037,
 0.00196337, 0.00219251, 0.00244362, 0.00245358, 0.00252321,
 0.00252963, 0.00289111, 0.002593, 0.00289984, 0.00272996, 0.00300527}
```

```
FittedModel[ 0.00125425 + 0.00011568 x ]
```



Z výše uvedeného obrázku je zřejmé, že srážková činnost v pozorované oblasti roste. Na ose x jsou zaznamenány jednotlivé obrázky, tedy 16 hodnot, na ose y k nim příslušné průměrné srážkové úhrny na jeden pixel v [mm/h]. Pro získaná data můžeme následně modelovat jejich vývoj v čase, bohužel ale s velikou mírou volatility.

7 Těžiště

Pro budoucí vývoj srážek je potřeba určit směr pohybu oblačnosti a jejich rychlost. Pro zjištění těchto dat se nám zpočátku jevila jako vhodná metoda použití těžiště, ale později jsme bohužel objevili její nedostatky. Pro výpočet těžiště použijeme transfomovaná data, kdy postupně vypočítáme pozice těžiště $[T_x, T_y]$ pro řádky a sloupce matice. Výpočty provedeme pro všech 16 nejaktuálnějších dat dle následujících vzorců:

$$T_x = \sum_{j=1}^{n}\sum_{i=1}^{m} x_{i,j} * j$$

$$T_y = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j} * i$$

```
souradnice = Table[, Dimensions[data1704172][[1]], 2];
For[i = 1, i ≤ Dimensions[data1704172][[1]], i++,
 souradnice[[i, 2]] =
  Total[Total[data1704172[[i]], {2}] * Table[j, {j, 375}]] / Total[data1704172[[i]], 2];
 souradnice[[i, 1]] = Total[Total[data1704172[[i]]] * Table[k, {k, 648}]] /
   Total[data1704172[[i]], 2];]
souradnice
```

```
{{233.522, 236.14}, {233.93, 238.881}, {238.076, 240.332}, {241.143, 239.429},
 {254.129, 237.508}, {250.463, 235.497}, {246.776, 234.904}, {247.67, 235.06},
 {250.191, 236.442}, {247.87, 241.173}, {264.345, 238.452}, {255.377, 240.387},
 {269.191, 242.067}, {256.938, 249.624}, {266.073, 253.263}, {274.205, 252.291}}
```

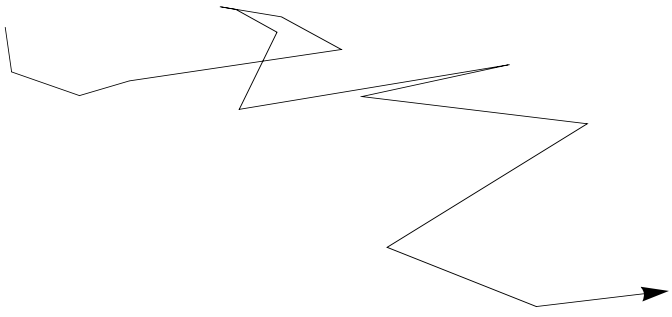
Abychom si mohli polohu těžiště znázornit graficky, přepíšeme druhé hodnoty těžišť (y-ové souřadnice) jejich opačnými hodnotami.

```
For[i = 1, i ≤ Dimensions[souradnice][[1]], i++,
souradnice[[i, 2]] = -souradnice[[i, 2]]]
souradnice
```

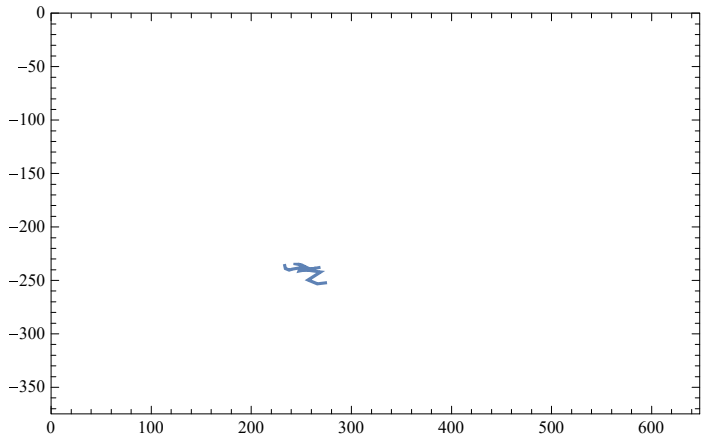
```
{{233.522, -236.14}, {233.93, -238.881}, {238.076, -240.332}, {241.143, -239.429},
 {254.129, -237.508}, {250.463, -235.497}, {246.776, -234.904}, {247.67, -235.06},
 {250.191, -236.442}, {247.87, -241.173}, {264.345, -238.452}, {255.377, -240.387},
 {269.191, -242.067}, {256.938, -249.624}, {266.073, -253.263}, {274.205, -252.291}}
```

Níže uvádíme graficky znázorněný pohyb těžiště a jeho znázornění k celé pozorované oblasti.

```
Graphics[Arrow[souradnice]]
```



```
ListLinePlot[souradnice, Frame → True, PlotRange → {{0, 648}, {0, -375}}]
```



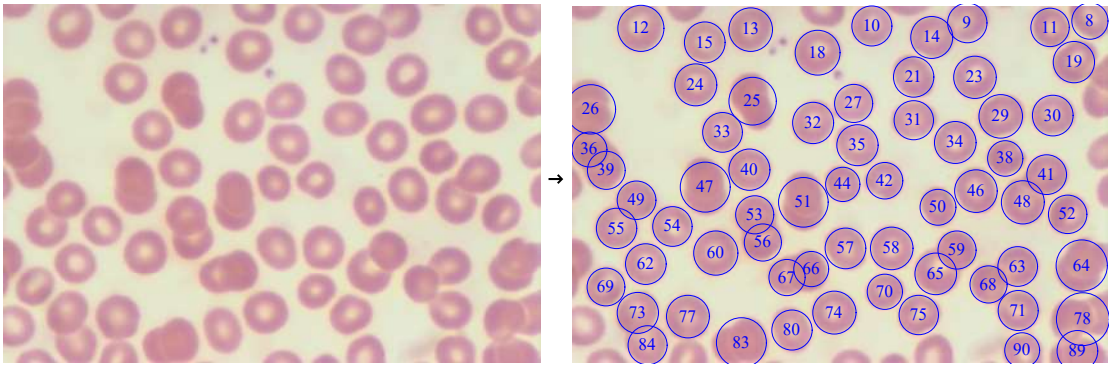
Ačkoliv je z importovaných dat, příp. z jejich animace (viz. kapitola 3) patrný výrazný posun srážek převážně východním směrem, těžiště dat se za 4 hodiny (16*15 minut) posunulo pouze o 40 km východně a 15 km jižně. Navíc, celý posun těžiště je spíše chaotický a nemá jasně definovaný trend. Bohužel k tomuto negativnímu výsledku dochází vstupováním nových srážek do pozorované oblasti a současným opuštěním oblasti jinými srážkami. V případě situace, že by jedna velmi významná srážková oblast vstoupila do pozorované oblasti a jiná velmi významná srážková oblast současně pozorovanou oblast opustila, došlo by k posunu těžiště opačným směrem, než ve směru pohybu srážek, i v rozmezí desítek kilometrů za krátký časový okamžik. Z tohoto důvodu nemůžeme,pro zkoumání pohybu srážek sledovat celou oblast najednou, ale je nutné jednotlivé srážkové oblasti identifikovat.

U tohoto způsobu jsme navíc uvažovali přiřadit jednotlivým hodnotám váhy, které by u okrajů obrázku nabývaly nižších hodnot a v centrální oblasti vyšších. Primárním záměrem bylo snížit vliv nově vstupujících hodnot. Bohužel by ale tato úprava vzniklý problém mohla částečně zmírnit, ale ne zcela odstranit, a proto bychom stále získávali zkreslené údaje.

8 How to Count Cells

Název této kapitoly je záměrně v angličtině a odkazuje na část názvu článku na blogu Wolfram uživatelky Shadi Ashnai, konkrétně na článek “How to Count Cells, Annihilate Sailboats, and Warp the Mona Lisa”. [10]

Paní Shadi Ashnai ve svém článku prezentuje možnost využití programu Wolfram *Mathematica* pro výpočet počtu buněk v obrázku, kdy kromě toho, že jednotlivě identifikuje téměř všechny buňky, navíc získá i informace o každé z nich, jako např. polohu a velikost. Grafickou prezentaci identifikaci buněk můžeme vidět níže. V našem případě použité příkazy lehce modifikujeme pro identifikaci jednotlivých srážkových oblastí.



Obrázek 8: Grafická prezentace identifikace buněk v projektu uživatele Shadi Ashnai.

Na začátku si zvolíme jeden datový obraz z přetransformovaných dat, na kterém si ukážeme postupné kroky. Na určení jednotlivých srážkových mraků nepotřebujeme znát intenzitu srážek, ale oblast výskytu, a proto je vhodné všem srážkám v oblasti přiřadit stejnou váhu. Nejjednoduššeji tohoto docílíme funkci Binarize, kterou místům s výskytem srážek přiradímě hodnotu 1 a místům bez výskytu srážek hodnotu 0. Následně pomocí funkce FillingTransform vyplníme ohraničené oblasti bez srážek, neboť se dá předpokládat, že v tomto místě prší slabě, resp. pršelo a bude pršet, a eliminujeme možnost, že se nám z důvodu tohoto prázdného prostoru srážková oblast v další identifikaci rozpadne na více částí. Tyto dvě funkce jsou definovány následovně:

Binarize [*image*] - vytvoří binární obraz způsobem, že hodnoty, které přesahují stanovenou hodnotu nahradí hodnotou 1, ostatní hodnotou 0.

Binarize [*image, t*] - vytvoří binární obraz způsobem, že hodnoty, které přesahují hodnotu *t,* nahradí hodnotou 1, ostatní hodnotou 0.

Binarize [*image,* $\{t_1, t_2\}$] - vytvoří binární obraz způsobem, že hodnoty, které jsou v rozmezí t_1 až t_2 nahradí hodnotou 1, ostatní hodnotou 0.

Binarize [*image, f*] - vytvoří binární obraz způsobem, že hodnoty, u kterých je funkce *f* [*v*] vyhodnocena jako pravdivá, nahradí hodnotou 1, ostatní hodnotou 0.

FillingTransform [*image*] - obrázek *image* je nahrazen jeho novou verzí, která má vyplněna všechna ohraničená minima.

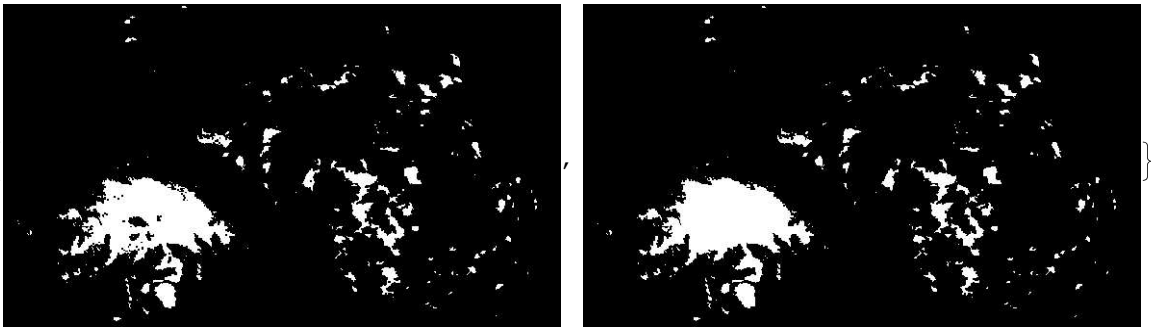
FillingTransform [*image, marker*] - obrázek *image* je nahrazen jeho novou verzí, která má vyplněna ohraničená minima definována obrázkem *marker* způsobem, že alespoň jeden korespondující element z obrázku *marker,* který náleží ohraničenému minimu v obrázku *image,* je nenulový.

FillingTransform [*image, h*] - obrázek *image* je nahrazen jeho novou verzí, která má vyplněna ohraničená minima, jejichž hodnota je rovna hodnotě *h* nebo menší.

```
htcc =
```

```
;
```

```
{Binarize[htcc], FillingTransform[Binarize[htcc]]}
```



V následujícím kroku si jednotlivé srážkové oblasti identifikujeme, resp. rozdělíme celý obrázek na jednotlivé oblasti. Na toto rozdělení použijeme funkce DistanceTransform, ImageAdjust, MaxDetect, GradientFilter a WatershedComponents. Provedené rozdělení si následně zobrazíme funkci Colorize.

Funkcí DistanceTransform nahradíme každý pixel zadaného obrázku jeho nejbližší vzdáleností od pozadí v daném obrázku. V případě, že tuto funkci použijeme např. na pravidelný kruh, největší hodnotu bude mít právě střed daného objektu. Pro tuto funkci jsou definovány následující zápisy:

DistanceTransform [*image*]

DistanceTransform [*image, t*] - pixely, které přesahují hodnotu *t* jsou považovány za popředí obrázku

Získaná data následně pomocí funkce ImageAdjust (zapisujeme: ImageAdjust[*image*]) znormujeme na rozsah od 0 do 1. Následně určíme lokální maxima pomocí funkce MaxDetect. Tuto funkci je v našem případě vhodné doplnit o hodnotu *h,* kterou rozšíříme rozsah hodnot pro lokální maximum, resp. za výsledek funkce budeme považovat i hodnotu v okolí lokálního maxima, která není menší

než hodnota lokálního maxima zmenšená o h. Funkce je definována následujícími zápisy:

MaxDetect [*image*]

MaxDetect [*image, h*]

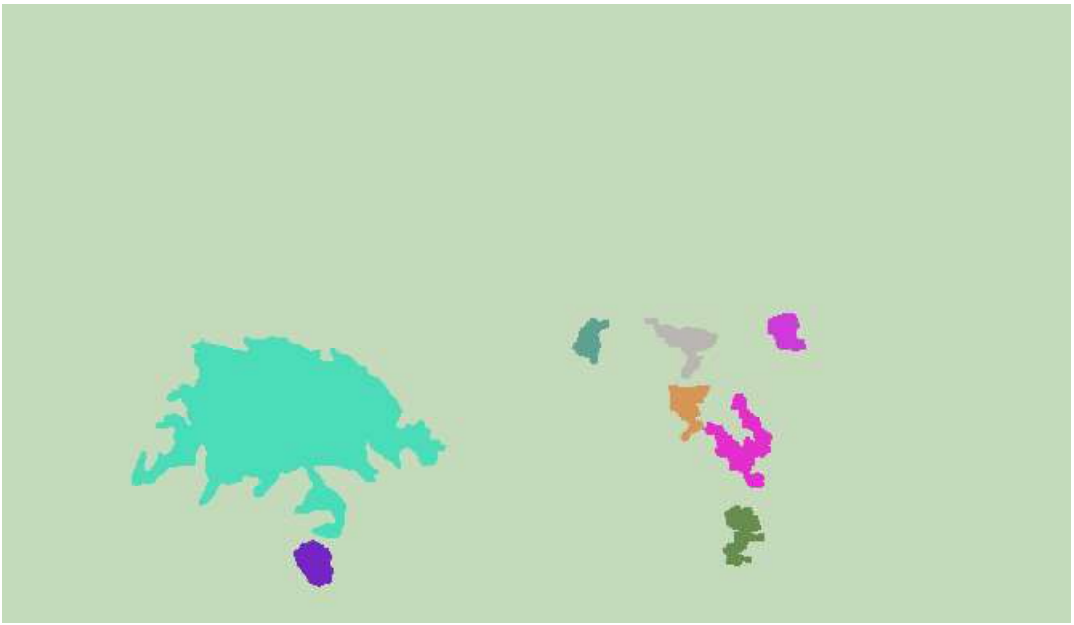
MaxDetect [*data,...*] - tuto funkci je možné použít i pro detekování maxim např.v matici

Pro konečnou identifikaci jednotlivých oblastí použijeme funkci WatershedComponents. Jak již vyplývá z názvu funkce, jedná se primárně o morfologickou funkci, která zadaný obrázek povrchu země rozděluje na jednotlivá povodí. Z tohoto důvodu je tedy pro náš účel vhodné jednotlivé srážkové oblasti ohraničit pomocí funkce GradientFilter. Tuto funkci zapisujeme ve tvaru GradientFilter [*image, r*], jejíž výsledkem je obrázek odpovídající velikosti gradientu zadaného obrazu, kdy jednotlivé body jsou vypočítávány pomocí diskrétní Gaussovy derivace (neboli Hermitovou funkcí) v poloměru *r* pixelů. Funkce WatershedComponents má následující formy zápisu:

WatershedComponents[*image*] - funkce vypočítá transformaci zadaného obrázku *image* a jejím výsledkem je pole kladných celých čísel. Každý pixel má přiřazeno právě jedno celé číslo, které udává oblast, do které náleží.

WatershedComponents [*image, marker*] - funkce využíví binární obrázek *marker*, který označuje oblasti, kde mohou být vytvořena jednotlivá pole.

```
b = FillingTransform[Binarize[htcc]];
distT = DistanceTransform[b];
marker = MaxDetect[ImageAdjust[distT], 0.15];
w = WatershedComponents[GradientFilter[b, 3], marker, Method → "Rainfall"];
Colorize[w]
```



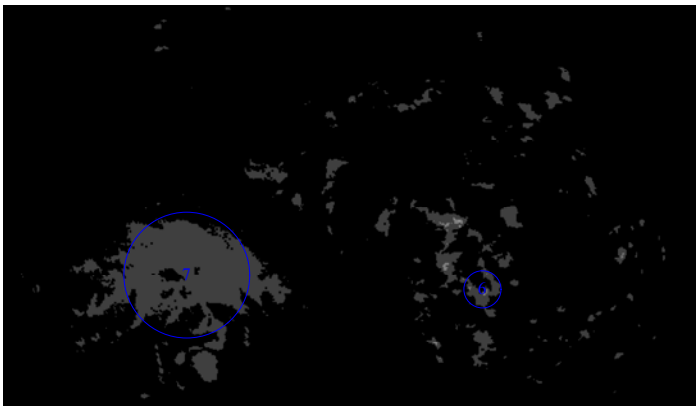
Autorka článku zvolila parametr *h* funkce MaxDetect na hodnotu 0.02. Jejím cílem bylo u dvou překrývajích se buněk nalézt právě 2 oblasti definují maxima a tím pádem poté pomocí funkce WatershedComponents identifikovat dvě buňky. V našem případě ale tento parametr využijeme opačným

způsobem, tedy jeho hodnotu nastavíme větší, konkrétně na 0.15. Při volbě větší hodnoty parametru právě naopak dvě (a více) úplně oddělené oblasti identifikujeme jako jednu větší, jelikož středy těchto oblastí propojíme a získáme z nich jeden objekt v obrázku marker. Negativním výsledkem většího parametru je eliminace menších objektů, které ale stejně v následujících krocích budeme zanedbávat. Vybrané srážkové oblasti můžeme zobrazit funkcí Colorize.

Jak můžeme vidět na obrázku výše, úpravou jsme získali 9 komponent. 8 komponent tvoří srážkové oblasti + 1 komponenta, která tvoří pozadí. Jelikož malé srážkové oblasti nejsou pro pozorování posunu nijak významné, zvláště proto, že u nich častěji než u větších může dojít k rozpadu, resp. vytvoření nové, je nutné zvolit pouze významné komponenty. K tomuto účelu nám poslouží funkce SelectComponents (zapisujeme SelectComponents[*m,"prop", crit*]), pomocí které zobrazíme pouze ty oblasti z datové množiny *m*, které splňují podmínku *crit*. Ostatní oblasti budou nahrazeny nulou. Následně pomocí funkce ComponentMeasurements (zapisujeme ComponentMeasurements [*m, "prop"*]) vypočítáme základní údaje zvolených oblastí.

```
cells = SelectComponents[w, "Count", 700 < # < 125000 &];
measures = ComponentMeasurements[cells, {"Centroid", "EquivalentDiskRadius", "Label"}]
Show[htcc, Graphics[{Blue, Circle @@ # & /@ (measures[[All, 2, 1 ;; 2]]),
   MapThread[Text, {measures[[All, 2, 3]], measures[[All, 2, 1]]}]}]]
```

```
{6 → {{448.981, 109.148}, 17.2793, 6}, 7 → {{171.89, 122.431}, 58.868, 7}}
```

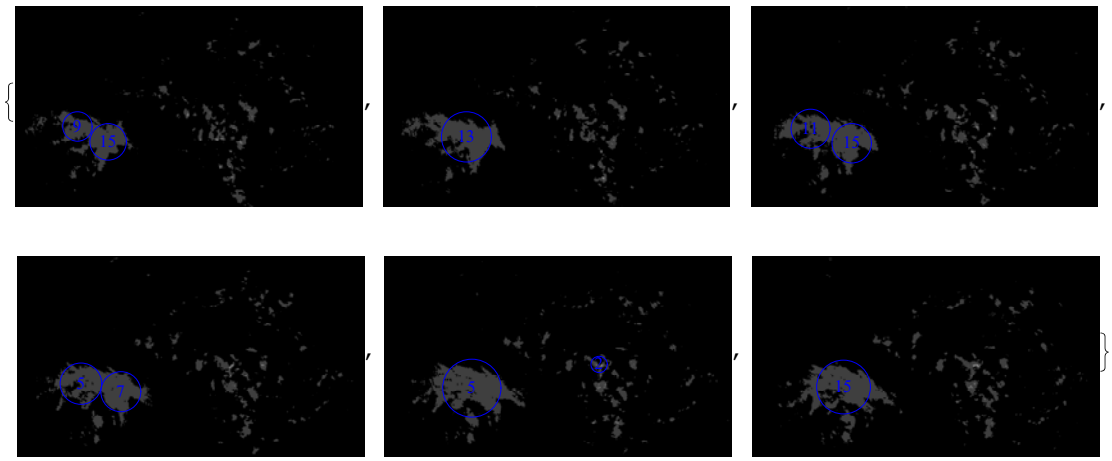


V případě, že zvolíme počet pixelů od 700 do 125 000 pixelů, identifikujeme objekty, jejichž velikost poloměru je přibližně 15 až 200 km.

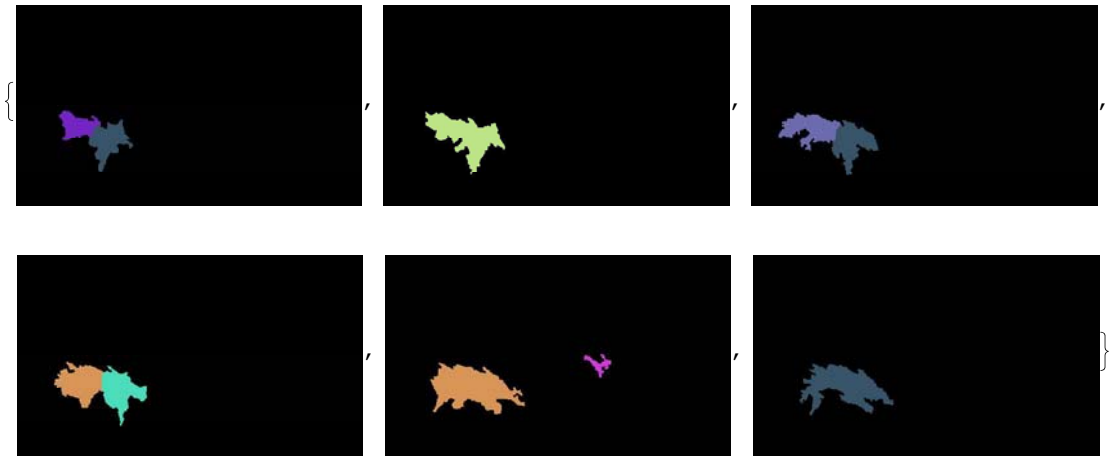
Nyní provedeme identifikaci pro všech 16 importovaných obrázků najednou a ve výstupu provedeme ukázku pro 6 obrázků.

```
b = Table[, Dimensions[data1704172][[1]]];
measures = Table[, Dimensions[data1704172][[1]]];
vystup = Table[, Dimensions[data1704172][[1]]];
obrcells = Table[, Dimensions[data1704172][[1]]];
For[i = 1, i ≤ Dimensions[data1704172][[1]], i++,
b[[i]] = FillingTransform[Binarize[Image[data1704172[[i]]]]];
distT = DistanceTransform[b[[i]]];
marker = MaxDetect[ImageAdjust[distT], 0.15];
 w = WatershedComponents[GradientFilter[b[[i]], 3], marker, Method → "Rainfall"];
Colorize[w];
cells = SelectComponents[w, "Count", 700 < # < 125000 &];
obrcells[[i]] = Colorize[cells];
measures[[i]] =
  ComponentMeasurements[cells, {"Centroid", "EquivalentDiskRadius", "Label"}];
vystup[[i]] = Show[Image[data1704171[[i]]],
   Graphics[{Blue, Circle @@ # & /@ (measures⟦i, All, 2, 1 ;; 2⟧),
     MapThread[Text, {measures⟦i, All, 2, 3⟧, measures⟦i, All, 2, 1⟧}]}]]]
```

```
Table[dataobr[[c]], {c, 6, 11}]
```



```
Table[obrcells[[c]], {c, 6, 11}]
```



Jak můžeme vidět na obrázcích výše, behem krátkého časového okamžiku může docházet k chybné identifikaci srážkových oblastí. V našem konkrétním případě, během 90 minut došlo k opako-

vanému “rozpadu” a “spojení” významné srážkové oblasti. Tudíž nemůžeme automaticky určit pozici, kam se daná oblast posunula, tedy rychlost a směr posunu. K tomoto jevu dochází z důvodu, že program méně pravidelné objekty identifikuje jako dva, které se překrývají, a proto pro náš případ není přístup uživatelky Shadi Ashnai až tolik vhodný, jak se na první pohled zdálo.

9 Morfologická analýza

Pojem morfologie pochází z řeckého slova morfé, které v překladu znamená podoba, případně tvar. Morfologická analýza se zabývá popisem povrchů těles, rostlin, minerálů, ale i popisem částí těl lidí a zvířat. Významnou částí morfologické analýzy je ale i její lingvistické uplatnění při zkoumání tvarosloví. Program *Mathematica* poskytuje v oblasti morfologické analýzy desítky funkci, pomocí kterých je možné provádět rozsáhlou analýzu obrazu. Navíc je možné tyto funkce kombinovat i se základními funkcemi zpracování obrazu a tím rozšířit možnosti jejich použití.

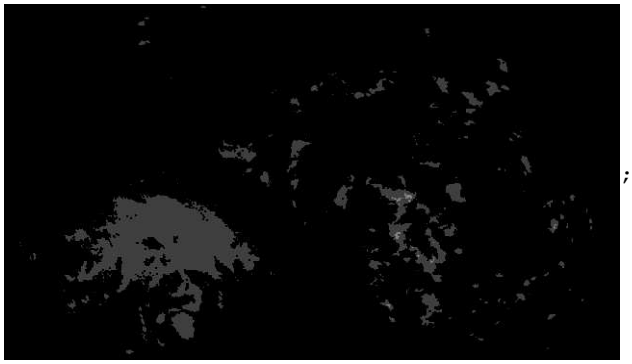
Jednou ze základních a pro nás klíčových funkci je funkce MorphologicalComponents. Tato funkce rozdělí původní obrázek na komponenty, resp. části, kdy každý pixel v obrázku je nahrazen celým číslem, které identifikuje komponentu, do které náleží. Jednotlivé komponenty jsou tvořeny souvislými částmi zdrojového obrázku a celá funkce ja zapisována následujícím způsobem:

MorphologicalComponents [*image*] - generuje pole kladných celých čísel. Každý pixel má přiřazeno právě jedno celé číslo, které reprezentuje propojenou oblast, do které náleží.

MorphologicalComponents [*image, t*] - pixely, které přesahují hodnotu *t* jsou považovány za popředí obrázku

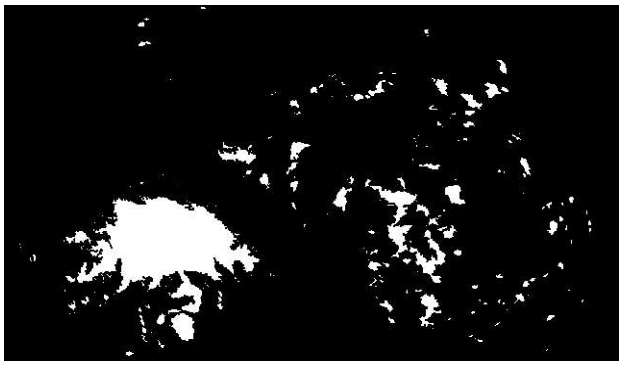
Před použitím samotné funkce je vhodné, stejně jako v předchozí kapitole, zdrojová data binarizovat a vyplnit ohraničená minima, abychom získali souvislé objekty.

```
ma =
```



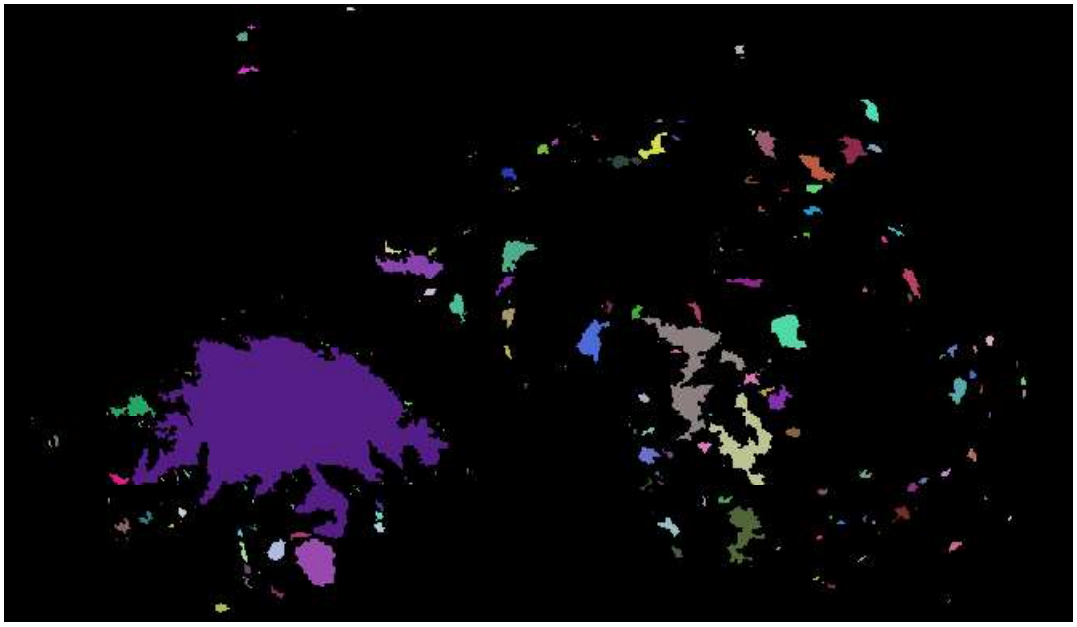
```
;
```

```
uprma = FillingTransform[Binarize[ma]]
```



```
mc = MorphologicalComponents[uprma];
```

```
MorphologicalComponents[uprma] // Colorize
```



```
cetnost = Reverse[SortBy[Tally[Flatten[mc]], Last]];

Dimensions[cetnost]

{225, 2}

cetnost[[Table[i, {i, 10}]]]

{{0, 225 053}, {85, 11 002}, {79, 1008}, {116, 615},
 {208, 492}, {183, 336}, {56, 328}, {77, 265}, {81, 222}, {51, 187}}
```

Jak můžeme vidět na obrázku výše, použitím funkce MorphologicalComponents na srážková data získáme obrovské množství komponent, jelikož každá souvislá oblast včetně jednopixelových tvoří samostatnou komponentu. Celkem tedy v našem konkrétním případě získáváme 224 samostatných komponent (0 nám sice ovlivňuje dimenzi, ale nejedná se o komponentu), z nichž významných není více jak 10. Za významné komponenty, stejně jako v předchozí kapitole, můžeme považovat takové, jejichž počet pixelů je alespoň 700, tedy v případě ideálního kruhového tvaru pokrývají oblast s poloměrem přibližně 15 km. Výše můžeme pozorovat 10 komponent s nejvyšší četností. Více jak 700 výskytů má hodnota 0, kterou ale nepovažujeme za komponentu, dále pak komponenta číslo 85 s četností 11 002 pixelů a komponenta číslo 79 s četností 1 008 pixelů. Menší komponenty jsou pro nás nedůležité především pro svoji "nestálost" v čase, tedy snadno vznikají i zanikají.

Pro eliminaci nevýznamných komponent a ponechání pouze těch významných můžeme použít další morfologickou funkci s názvem DeleteSmallComponents, jejíž zápis vypadá následovně:

DeleteSmallComponents [*image*] - nahradí malé propojené komponenty v binárním obrázku *image* pixely, kterými je generováno pozadí.

DeleteSmallComponents [*m*] - nahradí kladná celá čísla v matici *m* nulou, pokud je jejich počet malý.

DeleteSmallComponents [*image, n*] - nahradí malé propojené komponenty v binárním obrázku *image* pixely, kterými je generováno pozadí. Za malé propojené komponenty jsou považovány takové, jejichž počet prvků není větší než *n.*

DeleteSmallComponents [*m, n*] - nahradí kladná celá čísla v matici *m* nulou, pokud jejich počet není větší než *n.*

```
vyznkomp = DeleteSmallComponents[mc, 700];
Colorize[vyznkomp]
```

Nyní provedeme identifikaci komponent pro všech 16 importovaných obrázků najednou.

```
mcbfdsm = Table[, Dimensions[data1704172][[1]]];
col = Table[, Dimensions[data1704172][[1]]];
measures = Table[, Dimensions[data1704172][[1]]];
For[i = 1, i ≤ Dimensions[data1704172][[1]], i++,
bf = FillingTransform[Binarize[Image[data1704172[[i]]]]];
mcbf = MorphologicalComponents[bf];
mcbfdsm[[i]] = MorphologicalComponents[DeleteSmallComponents[mcbf, 700]];
col[[i]] = Colorize[mcbfdsm[[i]]];
measures[[i]] = ComponentMeasurements[mcbfdsm[[i]], {"Centroid", "Area"}];
]
col
```

Při pohledu na výše uvedené obrázky můžeme vidět, že identifikace objektů pomocí morfologické analýzy, přesněji pomocí funkce MorphologicalComponents**,** která rozděluje komponenty podle celých souvislých ploch, je jistě lepší volbou. Bohužel ale, v případě rozdělení srážkové oblasti na dvě samostané, dojde i k rozdělení komponent. Níže, v získané proměnné measures, vidíme základní statistiky ze získaný obrázků ve tvaru: n -> $\{\{s_1, s_2\},c\}$, kde hodnota n udává číslo komponenty v daném obrázku, hodnoty s_1, s_2 souřadnice středu a hodnota c množství pixelů dané komponenty. Souřadnice středu jsou zadány jako x_i, y_i, tedy jako souřadnice pro kartézskou soustavu souřadnic. V případě, že chceme získat hodnoty jako hodnotu řádku a sloupce datové matice, je možné provést

drobnou transformaci. Jelikož jsou ale pro nás na konci této kapitoly vhodná data v tomto tvaru, transformaci nyní provádět nebudeme.

```
measures // TableForm
```

```
1 → {{88.9306, 147.184}, 3248.88}    2 → {{144.484, 113.649}, 934.625}
1 → {{101.499, 144.42}, 3806.75}     2 → {{155.595, 115.799}, 904.75}     3 → {{398.673, 17
1 → {{112.629, 139.865}, 4116.38}    2 → {{168.652, 114.5}, 839.625}      3 → {{399.021, 18
1 → {{133.34, 132.748}, 5822.5}      2 → {{408.646, 16.3706}, 743.375}
1 → {{142.1, 131.213}, 5338.38}
1 → {{148.127, 130.928}, 6450.25}
1 → {{147.796, 131.218}, 7596.13}
1 → {{149.836, 131.609}, 8711.75}
1 → {{155.554, 128.944}, 9281.75}
1 → {{162.384, 127.272}, 9562.13}
1 → {{169.033, 128.512}, 8513.25}
1 → {{414.895, 154.079}, 1022.63}    2 → {{171.4, 122.333}, 11 054.3}
1 → {{171.601, 122.293}, 9157.88}
1 → {{178.573, 116.566}, 10 616.6}   2 → {{213.252, 42.5603}, 954.25}
1 → {{190.366, 113.112}, 10 151.4}   2 → {{227.634, 44.2519}, 1086.75}
1 → {{452.386, 171.234}, 838.}       2 → {{201.975, 110.425}, 10 240.5}   3 → {{241.77, 46.
```

Abychom mohli komponenty nadále zpracovávat, je vhodné provést jejich přetransformování na novou proměnnou *trans*. Získané hodnoty *measures* jsou bohužel pro další zpracování méně praktické, neboť hodnoty středů a velikost komponenty se nacházejí v jiných úrovních vnoření seznamu. Při této transformaci navíc provedeme znormování všech dat na rozsah ⟨0,1⟩. Pro znormování hodnot středu komponenty provedeme jejich vydělení počtem pixelů v daném rozměru, tedy souřadnici x_i počtem sloupců celého obrázku a souřadnici y_i počtem řádků celého obrázku. Množství pixelů vydělíme počtem pixelů celého obrázku, tedy hodnotou 243 000 (375*648). Znormované hodnoty jsou zobrazeny v tabulce níže.

```
trans = Table[, 16];
For[i = 1, i ≤ Dimensions[measures][[1]], i++,
trans[[i]] = Table[, Dimensions[measures[[i]]][[1]], 3];
For[j = 1, j ≤ Dimensions[measures[[i]]][[1]], j++,
trans[[i, j, 1]] = measures[[i, j, 2, 1, 1]] / 648;
trans[[i, j, 2]] = measures[[i, j, 2, 1, 2]] / 375;
trans[[i, j, 3]] = measures[[i, j, 2, 2]] / (375 * 648);
]]
TableForm[trans, TableHeadings → {{"1", "2", "3", "4", "5", "6", "7",
    "8", "9", "10", "11", "12", "13", "14", "15", "16"}, {"1", "2", "3"}}]
```

	1	2	3
1	0.137239	0.222969	
	0.392491	0.303065	
	0.0133699	0.00384619	
2	0.156634	0.240116	0.615237
	0.38512	0.308799	0.0453721
	0.0156656	0.00372325	0.00314969
3	0.17381	0.260266	0.615773
	0.372973	0.305333	0.0505748
	0.0169398	0.00345525	0.00302366
4	0.205772	0.630626	
	0.353996	0.0436549	
	0.0239609	0.00305916	
5	0.219291		
	0.349902		
	0.0219686		
6	0.228592		
	0.349141		
	0.0265442		
7	0.228081		
	0.349915		
	0.0312598		
8	0.231229		
	0.350958		
	0.0358508		
9	0.240052		
	0.343851		
	0.0381965		
10	0.250593		
	0.339392		
	0.0393503		
11	0.260853		
	0.342698		
	0.035034		
12	0.64027	0.264506	
	0.410878	0.326222	
	0.00420833	0.0454907	
13	0.264816		
	0.326114		
	0.0376867		
14	0.275576	0.329092	
	0.310843	0.113494	
	0.0436898	0.00392695	
15	0.293774	0.351287	
	0.301631	0.118005	
	0.0417752	0.00447222	
16	0.698127	0.311689	0.373102
	0.456623	0.294466	0.124568
	0.00344856	0.042142	0.00479064

Transformované hodnoty nyní porovnáme pomocí korelace a budeme zkoumat získané korelační koeficienty. Naším cílem je přitom identifikovat jednotlivé komponenty na různých obrázcích. Ty komponenty na dvou po sobě jdoucích obrázcích, jejichž charakteristiky (poloha středu a počet pixelů) mají nejvyšší korelaci, jsou zřejmě toutéž komponentou ve dvou po sobě následujících časových okamžicích. Pro použití korelace má program *Mathematica* definovanou funkci Correlation s následujícími zápisy:

Correlation [v_1, v_2] - výsledkem funkce je hodnota korelačního koeficientu pro zadané vektory v_1 a v_2.

Correlation [m] - výsledkem funkce je korelační matice pro matici m.

Correlation [m_1, m_2] - výsledkem funkce je korelační matice pro matice m_1 a m_2.

Touto funkcí zkoumáme všechny po sobě jdoucí dvojice komponent.

```
korelace = Table[, 15];
For[i = 1, i ≤ 15, i++,
korelace[[i]] = Table[, Dimensions[trans[[i]]][[1]], Dimensions[trans[[i + 1]]][[1]]];
For[j = 1, j ≤ Dimensions[trans[[i]]][[1]], j++,
For[k = 1, k ≤ Dimensions[trans[[i + 1]]][[1]], k++,
korelace[[i, j, k]] = Correlation[trans[[i, j]], trans[[i + 1, k]]];
]]]

TableForm[korelace,
 TableHeadings → {{"1-2", "2-3", "3-4", "4-5", "5-6", "6-7", "7-8", "8-9",
    "9-10", "10-11", "11-12", "12-13", "13-14", "14-15", "15-16"},
   {"komponenta 1 s ...", "komponenta 2 s ...", "komponenta 3 s ..."}}]
```

	komponenta 1 s ...	komponenta 2 s ...	komponenta 3 s ...
1-2	0.998106 0.875269 -0.135262	0.921835 0.998977 0.318248	
2-3	0.997726 0.867471 -0.0660681	0.930211 0.997003 0.36826	-0.00668126 0.431877 0.999968
3-4	0.991926 -0.0106469	0.947302 0.428297	0.128141 0.999928
4-5	0.998305	0.173855	
5-6	0.99961		
6-7	0.999952		
7-8	0.999996		
8-9	0.998969		
9-10	0.998836		
10-11	0.999472		
11-12	0.811679 0.9988		
12-13	0.843437	0.999971	
13-14	0.996697 0.611549		
14-15	0.99547 0.665848	0.740983 0.999943	
15-16	0.930127 0.996501 0.732142	0.931999 0.787984 0.999997	

V tabulce výše můžeme sledovat všechny získané korelační koeficienty po sobě jdoucích obrázků. Na prvním řádku, který je označený "1-2" se nacházejí korelace komponent mezi prvním a druhým obrázkem. Sloupce nám pak udávají korelace jednotlivých komponent z prvního obrázku s komponentami z druhého. Konkrétně např. hodnota ve druhém sloupci a prvním řádku (0.921835) nám uvádí korelační koeficient mezi druhou komponentou z prvního obrázku s první komponentou druhého obrázku.

Již při prvním pohledu na získaná data můžeme vidět, že komponenty srážkových oblastí, které na sebe navazují, jsou vzájemně vysoce korelované. Abychom tyto hodnoty oddělili od méně korelovaných, které na sebe nenavazují, je vhodné zvolit minimální hodnotu korelace. Pro naše účely dobře poslouží hodnota korelace 0.95. I přesto, že je tato hodnota velmi vysoká, může nastat případ, kdy toto kritérium splní dvě a více následující komponenty. Jelikož ale následovník může být pouze jeden, do programu navíc přidáme podmínku, že v případě více velmi vysokých korelačních koeficientů je následovníkem komponenta s vyšší mírou korelace.

Pro další zpracování již nemusíme znát přesné hodnoty korelačního koeficientu, a proto, v případě, že korelace dosahuje vyšších hodnot než 0.95 a současně je nevyšší hodnotou korelace mezi komponentami po sobě jdoucích obrázků, nahradíme tuto hodnotu hodnotou 1.

```
nasledovnici = Table[, Dimensions[korelace][[1]]];
For[i = 1, i ≤ Dimensions[korelace][[1]], i++,
nasledovnici[[i]] = Table[, Dimensions[korelace[[i]]][[1]]];
For[j = 1, j ≤ Dimensions[korelace[[i]]][[1]], j++,
nasledovnici[[i, j]] = Table[, Dimensions[korelace[[i, j]]][[1]]];
For[k = 1, k ≤ Dimensions[korelace[[i, j]]][[1]], k++,
If[And[korelace[[i, j, k]] > 0.95, Max[korelace[[i, j]]] == korelace[[i, j, k]]],
    nasledovnici[[i, j, k]] = 1, nasledovnici[[i, j, k]] = 0]
]]]
TableForm[nasledovnici,
 TableHeadings → {{"1-2", "2-3", "3-4", "4-5", "5-6", "6-7", "7-8",
    "8-9", "9-10", "10-11", "11-12", "12-13", "13-14", "14-15", "15-16"},
   {"komponenta 1 s ...", "komponenta 2 s ...", "komponenta 3 s ..."}}]
```

	komponenta 1 s ...	komponenta 2 s ...	komponenta 3 s ...
1-2	1	0	
	0	1	
	0	0	
2-3	1	0	0
	0	1	0
	0	0	1
3-4	1	0	0
	0	0	1
4-5	1	0	
5-6	1		
6-7	1		
7-8	1		
8-9	1		
9-10	1		
10-11	1		
11-12	0		
	1		
12-13	0	1	
13-14	1		
	0		
14-15	1	0	
	0	1	
15-16	0	0	
	1	0	
	0	1	

V tabulce výše je nyní přehledně vidět komponenty, které na sebe navazují. Např. na komponentu č.1 z prvního obrázku navazuje komponenta č.1 z druhého obrázku. Stejně tak na ní navazuje komponenta č. 1 z třetího obrázku apod. Graficky si toto můžeme ukázat na prních třech obrázcích níže. Stejně číslované kompononety jsou znázorněny stejnou barvou, konkrétně komponenta číslo 1 je znázorněna žlutou barvou, komponenta číslo 2 fialovou barvkou a komponenta číslo 3 zelenou barvou. Na obrázcích je možné vidět, že první (žlutá) komponenta má ve všech třech obrázcích přibližně stejnou polohu i tvar

```
Table[col[[c]], {c, 1, 3}]
```

Nyní, když dokážeme určit, které komponenty označují stejné srážkové oblasti v různých po sobě jdoucích obrázcích, využijeme tato data pro získání informací o směru a rychlosti pohybu dané srážkové oblasti. Navíc, jelikož samotné území České republiky, resp. samotná pozorovací oblast je velmi malá vzhledem k velikosti celých srážkových front, můžeme získaná data použít pro předpověď pohybu celé oblačnosti na našem území.

Jelikož nás zajímají pouze komponenty, které spolu korelují, pro další výpočty je pro nás vhodnější soustředit se pouze na ně. Pomocí funkce Position můžeme zobrazit pouze pozice s hodnotou 1, tedy údaje o na sebe navazujících komponentách.

```
pozice = Position[nasledovnici, 1]
```

```
{{1, 1, 1}, {1, 2, 2}, {2, 1, 1}, {2, 2, 2}, {2, 3, 3}, {3, 1, 1}, {3, 3, 2},
 {4, 1, 1}, {5, 1, 1}, {6, 1, 1}, {7, 1, 1}, {8, 1, 1}, {9, 1, 1}, {10, 1, 1},
 {11, 1, 2}, {12, 2, 1}, {13, 1, 1}, {14, 1, 1}, {14, 2, 2}, {15, 1, 2}, {15, 2, 3}}
```

Transforamcí získáváme trojici dat, ze kterých můžeme snadno vyčíst veškeré zjištěné informace. Konkrétně pozice 1 udává hodnotu prvního z dvojice obrázků, mezi nimiž pozorujeme korelované komponenty, přesněji např. u hodnoty 5 s jedná o dvojici obrázků 5 a 6. Na pozicích 2 a 3 poté nalezneme čísla korelovaných komponent v obrázcích dle pozice 1. Tedy např. z trojice {15,1,2} můžeme vyčíst, že komponenta číslo 1 v obrázku 15 je komponentou číslo 2 v obrázku 16 v čase o 15 minut později.

Nyní si pro nejvýraznější srážkovou oblačnost v našich datech vypočítáme postupně čísla komponent v získaných obrázcích a k nim příslušející středy.

```
a = 1;
b = {};
s = {measures[[1, 1, 2, 1]]};
For[i = 1, i <= 15, i++,
 p1 = Select[pozice, #[[1]] == i &];
 a = Flatten[Select[p1, #[[2]] == a &]][[3]];
 souradnice = measures[[i + 1, a, 2, 1]];
 s = Append[s, souradnice];
 b = Append[b, a];
]
b
s
```

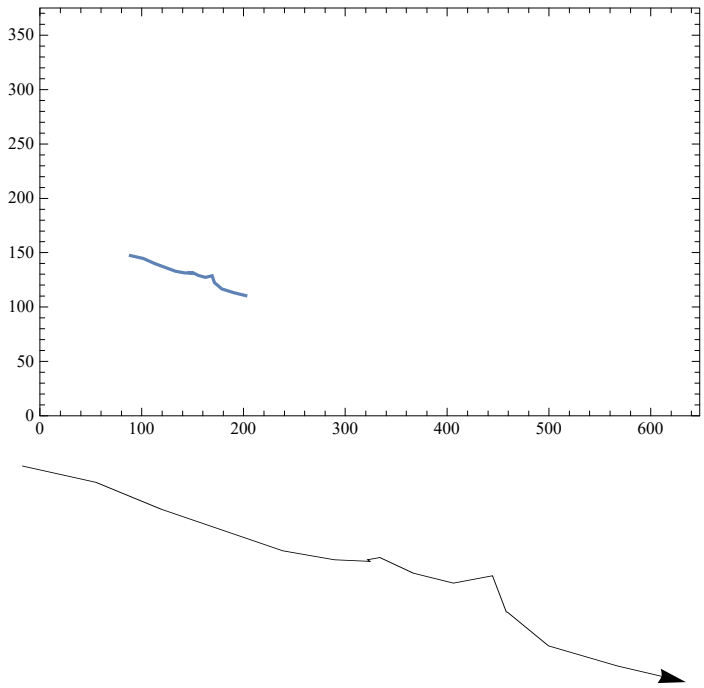
```
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2}
```

```
{{88.9306, 147.184}, {101.499, 144.42}, {112.629, 139.865}, {133.34, 132.748},
 {142.1, 131.213}, {148.127, 130.928}, {147.796, 131.218}, {149.836, 131.609},
 {155.554, 128.944}, {162.384, 127.272}, {169.033, 128.512}, {171.4, 122.333},
 {171.601, 122.293}, {178.573, 116.566}, {190.366, 113.112}, {201.975, 110.425}}
```

Pomocí tohoto programu hledáme mezi komponentami u po sobě jdoucích obrázků shodu následující komponenty u předcházejího obrázku a předcházející komponenty u obrázku následujícího. Např. u souřadnic {14,1,1} hledáme takovou trojici, která bude mít tvar {15,1,*}, tedy první pozice musí mít u následující trojice hodnotu o 1 vyšší, abychom vybírali pouze z následujících korelovaných dvojic. Pozice číslo 3 v trojici {14,1,1} musí odpovídat pozici číslo 2 v trojici {15,1,*},neboť popisují stejnou komponentu. * nám poté udává číslo komponenty v následujícím obrázku, konkrétně v našem případě se bude jednat o komponentu číslo 2.

Výstupem programu získáváme hodnotu *b*, která nám popisuje čísla jednotlivých komponent stejné srážkové oblasti v po sobě jdoucích obrázcích. Důležitější je ale pro nás hodnota *s.* Tato hodnota popisuje středy jednotlivých, po sobě jdoucích, komponent, ze kterých můžeme získat informace o rychlosti a směru pohybu oblačnosti. Tyto hodnoty získáváme ve tvaru x_i, y_i jakožto souřadnice kartézské soustavy souřadnic, a proto je můžeme bez jakékoliv úpravy vykreslit funkcí ListLinePlot.

```
g = ListLinePlot[s, Frame → True, PlotRange → {{0, 648}, {0, 375}}]
Graphics[Arrow[s]]
```




```
model = LinearModelFit[s, x, x]
```

FittedModel[177.333 – 0.321091 x]

Jak můžeme vidět na grafickém výstupu, získali jsme téměř lineární trend nejen v pohybu pozorované srážkové oblasti, ale i v pohybu celé srážkové oblačnosti na území České republiky (za značně zjednodušeného předpokladu, že pohyb oblačnosti je na tak malém území rovnoměrný), pomocí něhož můžeme předpovídat v krátkém časovém horizontu budoucí vývoj. Je vidět, že převládá pohyb oblačnosti ze západu na východ a ze severu na jih, přičemž druhá tendence je výrazně slabší, neboť absolutní hodnota odhadnuté směrnice je menší než 1.

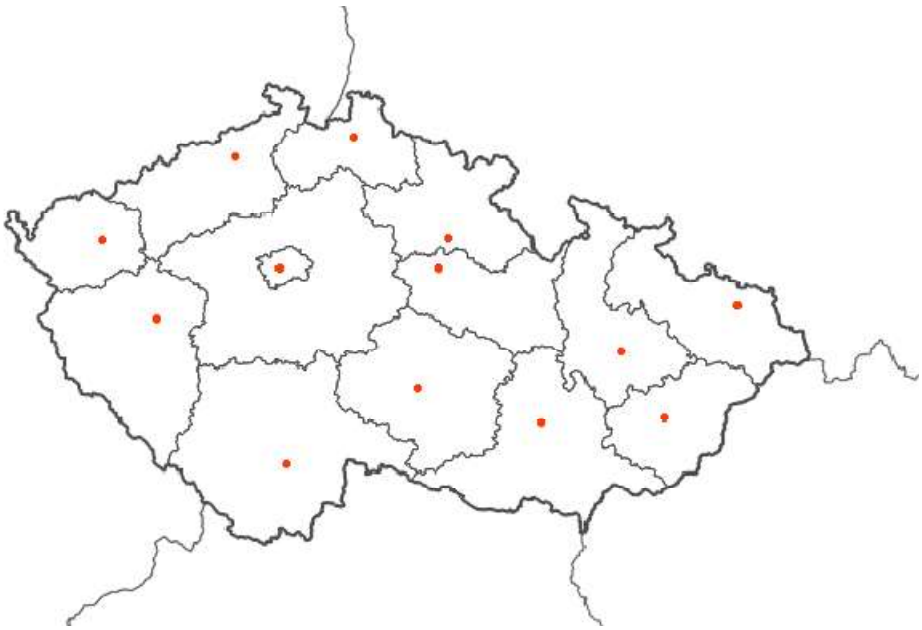
10 Kruhové okolí

Jelikož identifikace jednotlivých srážkových oblastí není vždy vzhledem k jejich proměnlivosti jednoznačná, jako v předchozí kapitole, můžeme k předpovědi srážek přistoupit poněkud z jiného úhlu pohledu. V případě, že si určíme jeden bod na mapě, můžeme sledovat množství srážek v jeho kruhovém okolí, tj. sledovat, zda v okolí daného bodu srážek přibývá, případně ubývá a na základě těchto dat krátce do budoucnosti předpovědět, zda bude v daném místě pršet.

Primárním záměrem bylo nadefinovat kruhová okolí pro všechny body na mapě současně, tedy uživatel programu by si zvolil jakýkoliv bod mapy, pro který by byla na základě kruhového okolí stanovena předpověď. Bohužel ale v případě volby okolí o poloměru 20 km program prováděl výpočet pro více jak 300 miliónů hodnot (konkrétně pro (648-20-20)*(375-20-20)*(20+20)*(20+20)=32 888 000 hodnot) u jednoho obrázku a výpočetní doba přesahovala 2 hodiny. Vzhledem k tomu, že výpočetní doba je více jak 8x delší než aktuálnost dat, je vhodné provádět výpočty pouze pro konkrétní bod na mapě. Pro jeden konkrétní bod v obrázku při stanovení stejného poloměru 20 km program provádí výpočet pouze pro 1 600 hodnot (konkrétně (20+20*(20+20)=1 600 hodnot) a výpočetní doba je menší jak 1 vteřina.

Na začátku jsme si určili souřadnice krajských měst; ve stejném měřítku, jako námi získaná data.

```
KV = {140, 153};
UnL = {90, 233};
L = {78, 305};
Pl = {188, 186};
Pr = {157, 260};
HK = {139, 362};
Pa = {157, 356};
CB = {276, 264};
J = {230, 344};
B = {251, 418};
Ol = {208, 467};
Os = {180, 537};
Z = {248, 493};
KrajskaMesta = {KV, UnL, L, Pl, Pr, HK, Pa, CB, J, B, Ol, Os, Z};
KrajskaMestaProGrafiku = Table[, 13, 2];
For[i = 1, i ≤ 13, i++,
KrajskaMestaProGrafiku[[i, 1]] = KrajskaMesta[[i, 2]];
KrajskaMestaProGrafiku[[i, 2]] = 375 - KrajskaMesta[[i, 1]];]
HighlightImage[podklad[[6]], KrajskaMestaProGrafiku]
```



Pro zvýraznění krajských měst na mapě můžeme použít např. funkci HighlightImage, pomocí které zvýrazníme body zadané pomocí souřadnic. Funkci je možné použít i na zvýraznění plochy, která je zadána binarizovaných obrázkem. Funkce má tedy následující dva tvary zápisu:

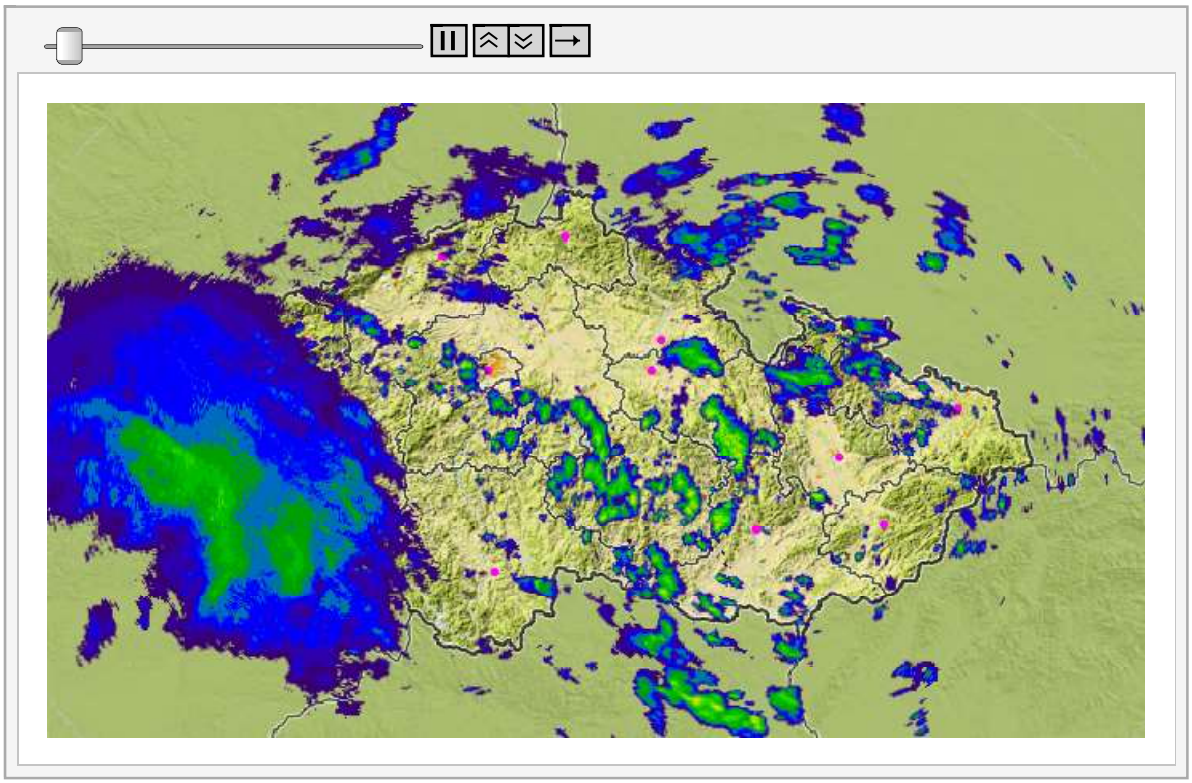
HighlightImage [*image,* $\{\{x_1, y_1\}, \ldots\}$] - výsledkem této funkce je obrázek, který obsahuje zvýrazněné pixely na pozici $\{x_i, y_i\}$

HighlightImage [*image, region*] - v obrázku *image* zvýrazní oblast, která je zadána obrázkem *region.*

Jelikož jsou jednotlivé souřadnice zadávány pomocí souřadnic x_i, y_i, bylo nutné provést drobnou transformaci zadaných poloh krajských měst, které definujeme jako bod pole, resp. matice pomocí

čísla řádku a sloupce. Získaný obrázek můžeme opět naanimovat pomocí funkce Animate včetně aktuální srážkové oblačnosti.

```
podklady = ImageCompose[ImageCompose[podklad[[3]], podklad[[4]]], podklad[[6]]];
podklady2 =
  HighlightImage[podklady, KrajskaMestaProGrafiku, "HighlightColor" → Magenta];
dataanimace = Table[ImageCompose[podklady2, zaznam[[i]]], {i, 1, Length[zaznam]}];
ListAnimate[dataanimace]
```

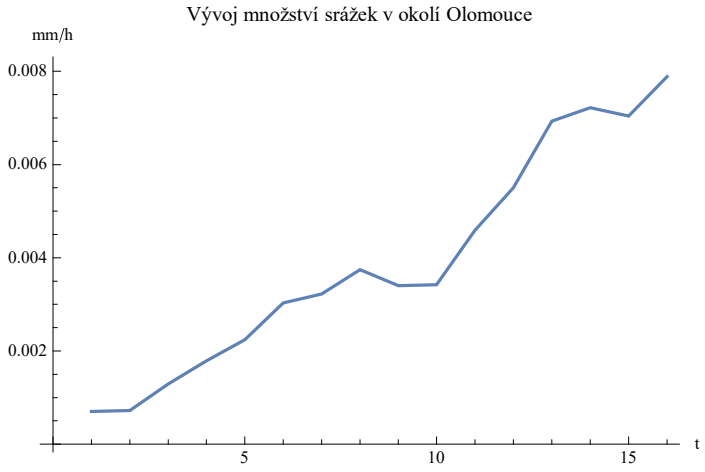


```
Export["animacesrazekskrajskymimesty.swf", %];
```

Jelikož prezentovanou animaci bohužel není možné vytisknout, její záloha s názvem *animacesrazekskrajskymimesty.swf* bude také přiložena jako příloha na CD.

Nyní si můžeme vypočítat jednotlivá množství srážek v okolí vybraného krajského města v různých časových okamžicích. Pro tento modelový příklad jsem zvolil nám blízké město Olomouc a hlavní město Prahu. Obě modelace jsou prováděny v kruhovém okolí 50 km od centra města. Program pro každý pixel ve "čtvercovém" okolí středu (u poloměru 50 zkoumáme 10 000 pixelů, 50 pixelů každým směrem ve směru osy x a ve směru osy y, přičemž střed s_1, s_2 je středem čtverce) zkoumá jeho euklidovskou vzdálenost od zadaného středu. Pokud se daný bod nachází v blízkosti menší než 50 km, započítá do proměnné *hodnota* internzitu srážek v daném bodě a do proměnné *pocet* započte jeho výskyt. Následně znormuje celkovou intenzitu srážek v oblasti počtem hodnot, čímž pro každý obrázek získáme průměrnou internzitu srážek pro každý bod v daném kruhovém okolí.

```
s1 = Ol[[1]];
s2 = Ol[[2]];
r = 50;
hodnota = Table[0, Dimensions[data1704172][[1]]];
pocet = Table[0, Dimensions[data1704172][[1]]];
hodpoc = Table[0, Dimensions[data1704172][[1]]];
For[k = 1, k ≤ Dimensions[data1704172][[1]], k++,
 For[i = s1 - r, i ≤ s1 + r, i++,
  For[j = s2 - r, j ≤ s2 + r, j++,
   If[Sqrt[(i - s1)^2 + (j - s2)^2] < r, And[pocet[[k]] = pocet[[k]] + 1,
     hodnota[[k]] = hodnota[[k]] + data1704172[[k, i, j]]],]
  ]]]
hodpoc = hodnota / pocet;
ListLinePlot[hodpoc, AxesLabel → {"t", "mm/h"},
 PlotLabel → "Vývoj množství srážek v okolí Olomouce"]
```



```
s1 = Pr[[1]];
s2 = Pr[[2]];
r = 50;
hodnota = Table[0, Dimensions[data1704172][[1]]];
pocet = Table[0, Dimensions[data1704172][[1]]];
hodpoc = Table[0, Dimensions[data1704172][[1]]];
For[k = 1, k ≤ Dimensions[data1704172][[1]], k++,
 For[i = s1 - r, i ≤ s1 + r, i++,
  For[j = s2 - r, j ≤ s2 + r, j++,
   If[Sqrt[(i - s1)^2 + (j - s2)^2] < r, And[pocet[[k]] = pocet[[k]] + 1,
     hodnota[[k]] = hodnota[[k]] + data1704172[[k, i, j]]],]
  ]]]
hodpoc = hodnota / pocet;
ListLinePlot[hodpoc, AxesLabel → {"t", "mm/h"},
 PlotLabel → "Vývoj množství srážek v okolí Prahy"]
```

Pokud se podíváme na získané grafy, na ose y se nachází průměrná intenzita srážek v daném kruhovém okolí v mm/h, na ose x nalezneme indexy jednotlivých obrázků. Index 1 odpovídá prvnímu časovému okamžiku, index 16 nejaktuálnějšímu pozorování. Jde tedy vlastně o časovou osu, kterou značíme písmenem t. Na grafu pro Olomouc můžeme pozorovou vzrůstající srážkový trend, tedy je zde vysoká pravděpodobnost, že v Olomouci se aktuálně srážky vyskutují a vyskytovat budou. Na grafu pro Prahu můžme pozorovat, že nejvyšších průměrných hodnot bylo dosaženo u dat s indexem číslo 12, následně pozorujeme výrazný pokles těchto hodnot. Můžeme tedy usuzovat, že v Praze již déšt ustává, resp. už ustal.

Při porovnání s morfologickou analýzou obrazu získáváme touto metodou málo vypovýdající data o budoucím vývoji. Tímto způsobem můžeme modelovat budoucí vývoj v řádu 1-2 hodin s velikou mírou volatility. Např. při zkoumání srážek v Olomouci sice můžeme předpovědět budoucí vývoj pomocí lineárního trendu, ale stejně tak víme, že množství srážek nemůže být stále rostoucí a tímto způsobem nemáme žádnou možnost odhadnout, kdy začne jejich množství klesat. Morfologickou analýzou ale naopak získáváme predikci pohybu celé srážkové oblačnosti. Limitováni jsme bohužel získanou oblastí, tedy nedokážeme predikovat srážky, které na území v budoucnosti vstoupí, ale i přes to jsme schopni, při příznivých vstupních podmínkách, predikovat vývoj na několik hodin

dopředu.

Závěr

Cílem mé diplomové práce bylo využít analýzu obrazu k předpovědi počasí, konkrétně importovat ze serveru http://pocasi.idnes.cz/ aktuální radarová data, analyzovat je a následně pomocí nich predikovat budoucí vývoj srážek na území České republiky. To vše pomocí programu Wolfram *Mathematica*.

Ačkoliv na mě toto téma zpočátku nepůsobilo obsáhle a napadla mě (i mého vedoucího Mgr. Ondřeje Vencáleka Ph.D.) spousta možností, jak provést predikci budoucích srážek, nebylo často možné tyto nápady využít. Možnosti často generovaly množství nových problémů, které se mi nepodařilo vyřešit. Přesto jsem nakonec ukázal několik cest, které by šly pro predikci použít - metodu těžiště, metodu morfologické analýzy, která je vylepšením metody výpočtu těžiště, a metodu kruhového okolí. Pomocí jednotlivých výsledků můžeme již nyní krátkodobě predikovat budoucí vývoj srážek, ale bohužel s velikou mírou volatility.

Celá práce pro mě byla velikým přínosem. Nejen z důvodu získání znalostí o dalším matematickém výpočetním programu, se kterým jsem dříve neměl možnost pracovat a který je nejobsáhlejší ze všech matematických programů, se kterými jsem měl možnost pracovat, ale zvláště pak z důvodu získání opravdu obrovského množství nových informací v oblasti zpracování obrazu. Tvorba této diplomové práce je pro mě přínosem do budoucího života, neboť mě naučila dívat se na problémy z několika úhlů pohledu a vytrvat v hledání nových řešení, když ta současná selžou.

Literatura

[1] VENCÁLEK, Ondřej. *Základy analýzy dat v softwaru Mathematica.* Olomouc: Univerzita Palackého v Olomouci, 2015. ISBN 978-80-244-4474-1.

[2] ŘÍHA, Jan. *Software Mathematica v přírodních vědách a ekonomii.* Olomouc: Univerzita Palackého v Olomouci, 2012. ISBN 978-80-244-2994-6.

[3] FRIEDRICH, Václav. *Mathematica na počítači pro nematematiky.* Ostrava: VŠB-TU Ostrava, 2013. ISBN 978-80-248-3162-6.

[4] HRON, Karel a Pavla KUNDEROVÁ. *Základy počtu pravděpodobnosti a metod matematické statistiky.* 2. vydání. Olomouc: Univerzita Palackého v Olomouci, 2015. ISBN 978-80-244-4774-2.

[5] *Srážky - Počasí - iDNES.cz* [online]. Praha 5: MAFRA, 2017 [cit. 2017-04-09]. Dostupné z: http://pocasi.idnes.cz/?t=img_s

[6] *Radarová síť ČHMÚ. Radarová síť ČHMÚ* [online]. Praha: Český hydrometeorologický ústav, (c)2010-2011 [cit. 2017-04-09]. Dostupné z: http://portal.chmi.cz/files/portal/docs/meteo/rad/info_czrad/index.html

[7] *TISKOVÁ ZPRÁVA: Úspěšná realizace projektu „Upgrade měřicích systémů pro předpovědní a výstražnou službu". In: TISKOVÁ ZPRÁVA: Úspěšná realizace projektu „Upgrade měřicích systémů pro předpovědní a výstražnou službu"* [online]. Praha, 2015, s. 1-10 [cit. 2017-04-09]. Dostupné z: http://portal.chmi.cz/files/portal/docs/ruzne/Radary/radary.pdf

[8] *Radarové snímky - nápověda — In-počasí* [online]. Plzeň: InMeteo, 2017 [cit. 2017-04-09]. Dostupné z: http://www.in-pocasi.cz/radarove-snimky/napoveda/

[9] *Ruseni meteorologickych radiolokatoru CZRAD* [online]. Praha 9: Český telekomunikační úřad, 2016 [cit. 2017-04-27]. Dostupné z: http://radar4ctu.bourky.cz/Ruseni.html

[10] *How to Count Cells, Annihilate Sailboats, and Warp the Mona Lisa—Wolfram Blog. Wolfram Blog: News and Ideas from Wolfram Research* [online]. Oxfordshire: Shadi Ashnai, 2012 [cit. 2017-04-20]. Dostupné z: http://blog.wolfram.com/2012/01/04/how-to-count-cells-annihilate-sailboats-and-warp-the-mona-lisa/

[11] *38826007.jpg. In: Panoramio - Help - Copy Photos to Google Maps* [online]. Mountain View, Kalifornie, USA: Google.com, 2017 [cit. 2017-04-09]. Dostupné z: http://static.panoramio.com/photos/original/38826007.jpg

[12] *Full_1820ac_Radar-2.JPG. In: Česká republika — Turistika.cz* [online]. Praha 1: Turistika.cz, 2017 [cit. 2017-04-09]. Dostupné z: https://foto.turistika.cz/foto/42547/32341/full_1820ac_Radar-2.JPG