



ZÁPADOČESKÁ  
UNIVERZITA  
V PLZNI



# Wanted: Shadows

aneb zjišťování výšky mraků pomocí Coralu a RaspberryPi

**M. Matta, E. Plic, T. Rajchman, D. Němec, L. Kohoutková**

**[astrox@mglzen.cz](mailto:astrox@mglzen.cz)**

Tým AstroX z Masarykovo Gymnázia

15. března 2023

# Osnova

- Kdo jsme?
- V čem soutěžíme?
- Jak jsme na soutěž přišli?
- Co jsme vymysleli?
- Jak jsem se dostali na FAV?
- Co jsme naprogramovali?
- Co se stane, až to doběhne?
- Poděkování
- Zdroje

# Poděkování

# Testovací snímek

# Kdo jsme?

## Středoškoláci z Masarykova gymnázia

- Lucie Kohoutková
- Matyáš Matta
- David Němec
- Eduard Plic
- Tomáš Rajchman

Monday, 19 Sept • 16:19

Nějaký nápad na název týmu na tu soutěž? Máš na to 30 sekund



AstroX

# V čem soutěžíme?



- mezinárodní soutěž Astro Pi pořádaná **Evropskou vesmírnou agenturou**
- mise Space Lab, která umožňuje týmům mladých studentů provést vlastní experiment na **ISS**
- programování mikropočítače Raspberry Pi programem napsaným v **Pythonu**

# Jak jsme na soutěž přišli?

- doporučení od kamaráda
- nevěděli jsme, kdy se bude konat, jelikož nepřicházely žádné další informace
- na poslední chvíli jsme **14. října** vyrazili do Prahy v narychlo sestaveném týmu

planetum



# Jak soutěž probíhala?

- prohlídka planetária
- vysvětlení soutěže
- vymýšlení nápadů
- večerní umělecké promítání
- vytváření prezentace
- prezentování
- vyhlášení





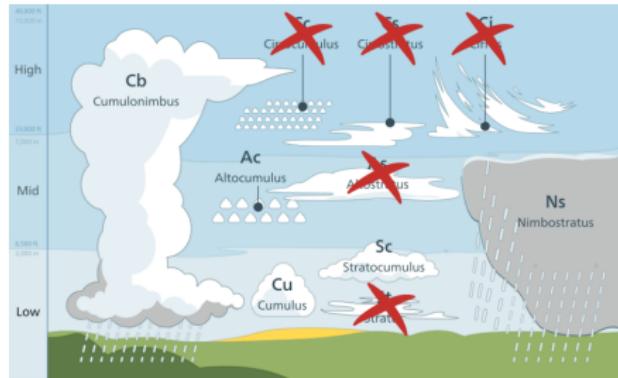
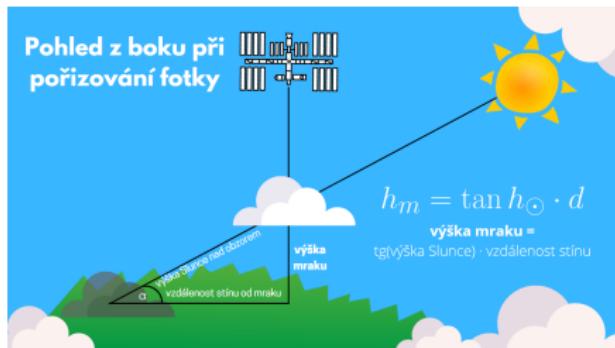
Obrázek: Nejvíce nás potěšila chálka<sup>1</sup>

---

<sup>1</sup>chálka = potrava, jídlo

# Co jsme vymysleli?

- spoustu nerealizovatelných nápadů
  - rozpoznání krajiny
  - velikost měst v závislosti na světle
- určování **výšky mraku**
  - s pomocí stínu
  - možné určení typu mraku
- nutný záložní plán



# Prezentace nápadu v Praze

- prezentace před tříčlennou porotou



## Pokročilá analýza výšky oblačnosti

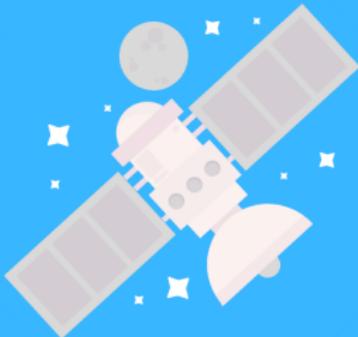
by AstroX

L. Kohoutková

M. Matta

E. Plic

T. Rajchman



# Prezentace nápadu

## Praktické využití systému



Levné a efektivní určování výšky a typu mraků



Letecká doprava



Meteorologie



# Prezentace nápadu v Praze



# Výhra v Praze

- z 6 týmů jsme obsadili **1. místo**



# Jak jsme návrh odeslali?

- museli jsme připravit text a odpovědět na zadané otázky

What type of data does your team plan to gather? How will this data help test your hypothesis?

Please tell us how you will use the Coral machine learning accelerator in your experiment.

# Odpověď'

- náš návrh byl v prosinci **schválen**

## Team members

Eduard Plic, 16  
Matyáš Matta, 16  
Tomáš Rajchman, 18  
Lucie Kohoutková, 14  
David Némec, 17

[Edit team members](#)

Time to submit your team's experiment for Phase 2 before 24 February 2023. See feedback from the Astro Pi team to help make the idea even better.



## Phase 1 feedback

Look at the skyfield python library, with this you can compute ISS location and the angle of solar elevation at the point below the ISS. This is a great idea, very excited to hear about the experiment after it runs on the ISS!

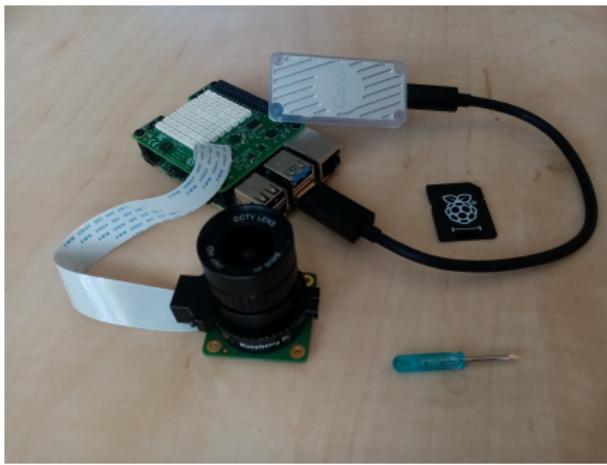
The [Astro Pi Mission Space Lab Phase 2 guide](#) guide covers Phase 2 for both Mission Space Lab themes: Life on Earth and Life in Space. We want your experiment to run reliably on the ISS, and this guide will help you get started quickly and give you the best chance of running your program without problems.

Even if you've entered the Astro Pi competition before, please make sure that you read and follow this guide, as many things have changed in comparison to previous years.

[Submit your team's experiment](#)

# Obdržení sady

- poté jsme obdrželi sadu **Raspberry Pi** s příslušenstvím až z **Nizozemska**



# Proč jsme potřebovali být na FAV?

- ve škole nebylo možné žádnou učebnu vyblokovat na 8 hodin
- školní **internet** (studenti MG ví)
- těžká koordinace projektu z domova

# Jak jsme se na FAV dostali?

- napsali jsme email doporučenému kontaktu podle stránek ZČU
- dostali jsme odpověď od doc. Vášy
- nápad jsme představili doc. Vášovi a doc. Masopustovi
- domluvili jsme se a dostali jsme **laboratoř** s vybavením

# Jak detektovat mraky?

# Použít RS-Net<sup>3</sup> deep learning

- po přečtení několika studií o RS-Net [3] architektuře jsme mysleli, že by se tento kód dal použít
- metoda postupně se zmenšujících bounding boxů <sup>2</sup>
- pro komplikovanost jsme nebyli schopni tuto metodu aplikovat

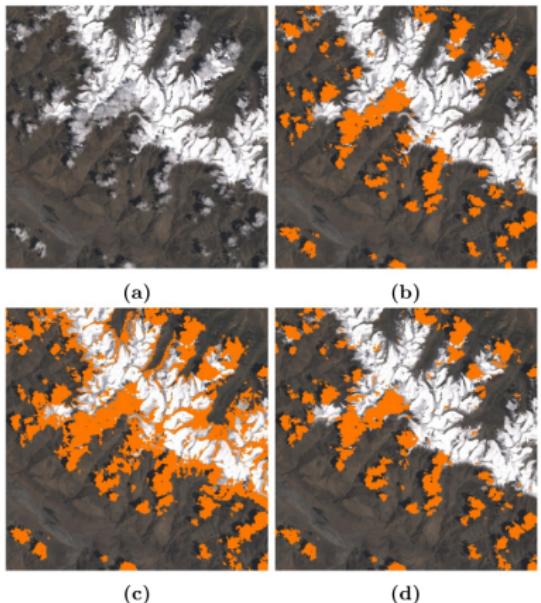


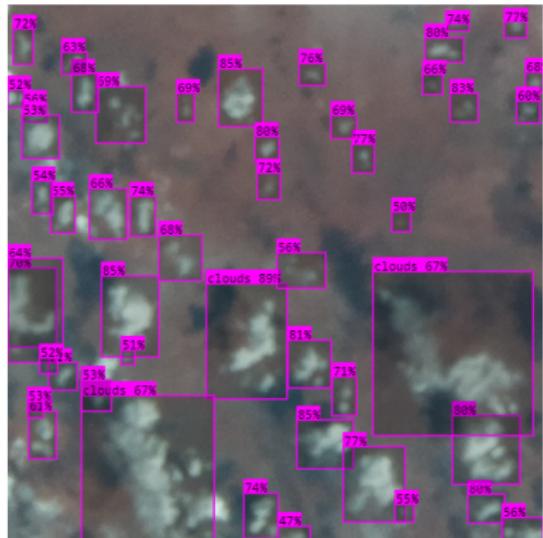
Fig. 5. Example of a SPARCS scene (LC81480352013195LGN00\_32) classified using Python Fmask and RS-Net, showing (a) RGB scene, (b) ground truth, (c) Fmask prediction and (d) RS-Net single band prediction.

<sup>2</sup>čtveřice souřadnic ohraničující objekt, reprezentován čtvercem

<sup>3</sup>Remote Sensing Network

# Bounding box umělá inteligence

- mohli jsme vytrénovat jednoduchý model na detekci mraků a stínů
- poté by stačilo udělat jednoduchý výpočet a zjistili bychom výšku
- detekce stínů se ukázala být příliš komplikovaná a nespolehlivá
- detekce mraků touto metodou se však ukázala být nejjednodušším řešením, takže jsme s ní dále pracovali



# Jak se trénuje „umělá inteligence“?

# Co je to umělá inteligence?

- poslední dobou spíše catchword<sup>4</sup>
- přesnějším popisem našeho využití je **strojové učení**
- modelu je předložen dataset<sup>5</sup> obsahující velké množství informací
- model v datech postupně začne hledat spojitosti a závislosti
- při dostatečném množství trénovacích dat je pak model schopen nalézt výsledky i na obrázcích, které neobsahují anotace<sup>6</sup>



---

<sup>4</sup> slovo, které vyvolává speciální pozornost

<sup>5</sup> soubor obrázků obsahující anotované<sup>6</sup> objekty

<sup>6</sup> soubory souřadnic ohraničující objekty

# Jak se připravuje dataset?

- použili jsme **Roboflow**<sup>7</sup>
- sehnali jsme obrázky z minulého ročníku Astro Pi
- vybrali jsme 16 obrázků (pozor na overfitting<sup>8</sup>)
- rozřezali jsme je na 16 sektorů
- ručně jsme udělali bounding box anotace



---

<sup>7</sup> online služba umožňující kolaboraci na anotaci společného datasetu

<sup>8</sup> trénování modelu na příliš ideálních datech

1940 px



485 px

# Jak jsme získali model?

- použili jsme Jupyter notebook<sup>9</sup> podle YouTube videa [2]
- vytrénovali jsme Tensorflow<sup>10</sup> 1.x model a překonvertovali jsme jej na kvantovaný **TFLite** model pro Coral<sup>11</sup>
- použili jsme Google Colab<sup>12</sup> kvůli velkému výkonu
- exportovali jsme Roboflow data do .tfrecord a .xml souborů
- manuálně jsme vytvořili labelmap
- model jsme trénovali několik hodin

---

<sup>9</sup> typ souboru kombinující snipetty Python kódu a textů k instrukcím

<sup>10</sup> bezplatná knihovna a platforma pro trénování modelů od Googlu

<sup>11</sup> nám poskytnuté příslušenství k Raspberry Pi pro zrychlení běhu AI modelů

<sup>12</sup> služba poskytující bezplatný hosting Jupyter notebooků

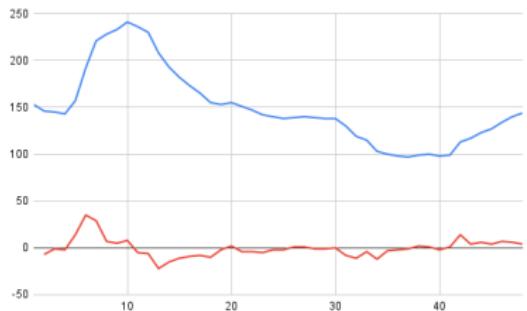
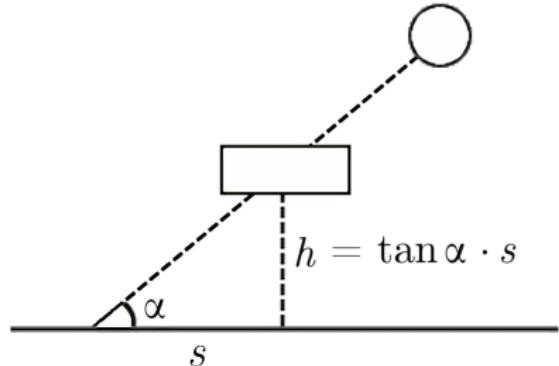
# Jak jsme model implementovali?

## Praktická ukázka kódu třídy ai

# Jak detektovat stíny?

# Cloud-shadow fingerprint

- svítivost jednotlivých pixelů ve směru, kde by se měl nacházet stín
- bod s nejvyšší svítivostí by byl mrak a bod s nejnižší svítivostí by byl stín



# Jakým směrem hledat stíny?

- stíny budou padat podle azimutu Slunce nad obzorem
- azimut Slunce je závislý na tom, kde je sever
- sever **není** fixní a nedá se z fotky určit
- Raspberry sice má magnetometr, ale pro kvalitní určení toho, kde je stín, by se odchylky musely pohybovat v jednotkách stupňů
- ...alespoň to tvrdila ESA

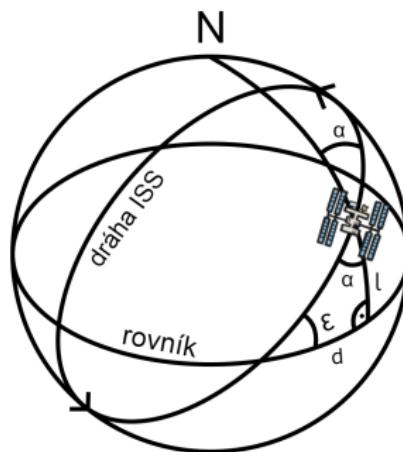
# Jak zjistit sever?

# Proč ne magnetometr?

■ ...

# Sever vůči ISS

- dá zjistit pomocí sférického trojúhelníku



, kde  $\alpha$  je úhel mezi severem a směrem pohybu ISS,  $\varepsilon$  je sklon dráhy ISS vzhledem k rovníku,  $l$  je zeměpisná šířka a  $d$  je pomocná proměnná, něco jako zeměpisná délka

# Sever vůči ISS

- použijeme sférickou sinová větu

$$\frac{\sin(d)}{\sin(\alpha)} = \frac{\sin(l)}{\sin(\varepsilon)} \Rightarrow \sin(d) = \frac{\sin(l)}{\sin(\varepsilon)} \sin(\alpha)$$

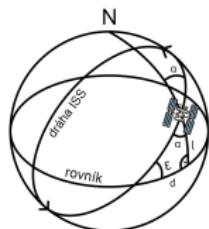
- kotangentový vzorec pro čtyři parametry:

$$\cos(90^\circ) \cos(d) = \cot(l) \sin(d) - \cot(\varepsilon) \sin(90^\circ)$$

$$\cot(\varepsilon) = \cot(l) \sin(d)$$

- zkombinujeme

$$\cot(\varepsilon) = \cot(l) \frac{\sin(l)}{\sin(\varepsilon)} \sin(\alpha) \Rightarrow \sin(\alpha) = \frac{\cos(\varepsilon)}{\cos(l)}$$



# Quo vadis ISS

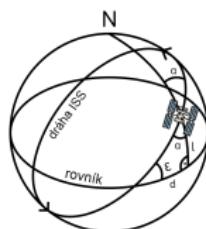
- konečná závislosť  $\alpha$  na zeměpisné šířce

$$\alpha = \arcsin \left( \frac{\cos(\varepsilon)}{\cos(l)} \right)$$

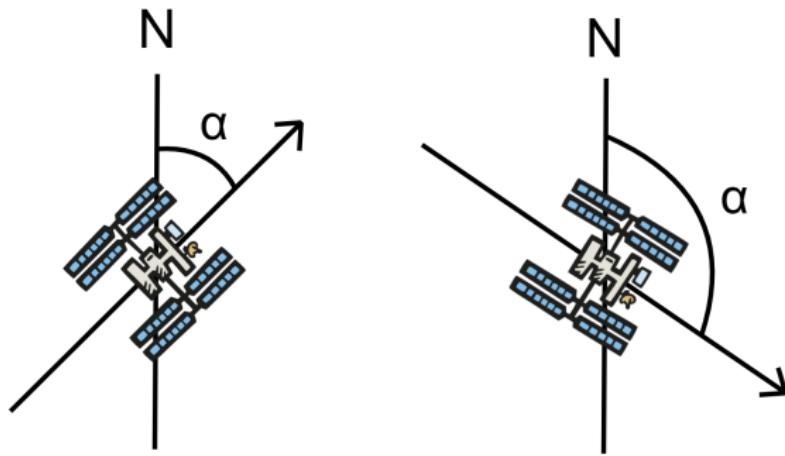
- vzorec vrátí ostrý úhel  $\rightarrow$  nutná korekce  $\alpha$ :

- **stoupající** šířka - úhel **zůstává**
- **klesající** šířka - vezme se jeho **doplňek ke 180°**

**NUTNÉ OŘÍZNOUT DOBŘE TEN DEBILNÍ OBRÁZEK NEBO NEVÍM PROČ SE HÝBE**



# Sever vůči ISS



- směr pohybu na fotce → směr severu z finální rovnice → směr stínu z azimutu Slunce

# Jak detektovat stíny?

# Jak projet jednotlivé pixely?

1. vypočteme střed mraku podle dat z třídy ai

```
def calculate_cloud_data(counter_for_shadows):  
    x_max = data[counter_for_shadows]['xmax']  
    y_max = data[counter_for_shadows]['ymax']  
    x_min = data[counter_for_shadows]['xmin']  
    y_min = data[counter_for_shadows]['ymin']  
    x_centre_of_cloud = (x_min+x_max)/2  
    y_centre_of_cloud = (y_min+y_max)/2  
    x_centre_of_cloud = round(x_centre_of_cloud, 0)  
    y_centre_of_cloud = round(y_centre_of_cloud, 0)  
    x_centre_of_cloud = int(x_centre_of_cloud)  
    y_centre_of_cloud = int(y_centre_of_cloud)  
    x_cloud_length = abs(x_max - x_min)  
    y_cloud_length = abs(y_max - y_min)
```

# Jak projet jednotlivé pixely?

- rozdělíme informaci o úhlu na kvadrant a základní úhel

## Směr kvadrantu a úhlu

Úhel je počítán po směru hodinových ručiček od severu (i.e. od vrchní strany), kvadranty jsou definovány obdobně, je to tedy jinak než je běžná matematická definice.

```
if 0 <= angle <= 90:  
    q = 1  
    angle_final = angle  
if 90 < angle <= 180:  
    q = 2  
    angle_final = angle - 90  
if 180 < angle <= 270:  
    q = 3  
    angle_final = angle - 180  
if 270 < angle <= 360:  
    q = 4  
    angle_final = angle - 270
```

# Jak projet jednotlivé pixely?

3. úhel přepočteme na  
přírůstek  $x$  a přírůstek  
 $y$
- $\alpha = 22.5^\circ$   
 $x += 0.38$   
 $y += 0.92$

```
angle_radians = np.radians(angle)
x_increase_meta = np.sin(angle_radians)
y_increase_meta = np.cos(angle_radians)
y_increase_meta = -y_increase_meta
x_increase_meta = np.round(x_increase_meta,5)
y_increase_meta = np.round(y_increase_meta,5)
x_increase_meta_abs = abs(x_increase_meta)
y_increase_meta_abs = abs(y_increase_meta)
```

# Jak projet jednotlivé pixely?

4. přírůstky přepočteme tak, aby (animace)  
alespoň jeden z nich byl 1

- $x += 0.38, y += 0.87$
- $x += 0.41, y += 1$

```
# make at least one variable 1 and the other smaller than 1
if x_increase_meta_abs > y_increase_meta_abs:
    x_increase_final = 1
    y_increase_final = y_increase_meta_abs/x_increase_meta_abs
if x_increase_meta_abs == y_increase_meta_abs:
    x_increase_final = 1
    y_increase_final = 1
if x_increase_meta_abs < y_increase_meta_abs:
    x_increase_final = x_increase_meta_abs/y_increase_meta_abs
    y_increase_final = 1
if angle_final == 0:
    x_increase_final = 0
    y_increase_final = 1
```

# Jak projet jednotlivé pixely?

## 5. vypočteme tzv. limit hledání stínů

```
# automatic limit calculation
# here we set basically how far we should search for shadows
sun_altitude_for_limit = shadow.sun_data.altitude(latitude, longitude, year, month,
day, hour, minute, second)
sun_altitude_for_limit_radians = sun_altitude_for_limit*(np.pi/180)
limit_cloud_height = 12000 #meters
limit_shadow_cloud_distance = limit_cloud_height/np.tan
(sun_altitude_for_limit_radians)
limit_shadow_cloud_distance_pixels = limit_shadow_cloud_distance/126.48
limit = limit_shadow_cloud_distance_pixels
```

# Jak funguje třída sun\_data?

```
def altitude(coordinates_latitude, coordinates_longitude, year, month, day, hour, minute, second):
    # use the NASA API to be able to calculate sun's position
    ts = api.load.timescale()
    ephem = api.load("ftp://ssd.jpl.nasa.gov/pub/eph/planets/bsp/de421.bsp")

    # define sky objects
    sun = ephem["Sun"]
    earth = ephem["Earth"]

    # given coordinates calculate the altitude (how many degrees sun is above the
    # horizon), additional data is redundant
    location = api.Topos(coordinates_latitude, coordinates_longitude, elevation_m=500)
    sun_pos = (earth + location).at(ts.tt(year,month,day,hour,minute,second)).observe
    (sun).apparent()
    altitude, azimuth, distance = sun_pos.altaz()
    altitude= float(altitude.degrees)
    return(altitude)
```

# Jak projet jednotlivé pixely?

- pro každý sektor je loop<sup>13</sup> definován znova
- máme tři situace,  $x > y$ ,  $x = y$  a  $x < y$
- celkem máme 12 (4 sektory, 3 situace) možností, jak kód může běžet
- kvůli komplikacím s univerzálním řešením jsme nakonec definovali každou situaci zvlášť

```

while True:
    # this is the count we use to run pixel by pixel
    count += 1
    if x_increase_final_abs < y_increase_final_abs:

        # check if x_sum is bigger than 1
        y_sum = abs(y_sum)
        if x_sum >= 1:
            x_sum -= 1
            x += 1

        # read pixel value
        data = (pix[x,y])

        # save the value, also search in only red
        value = round(average(data))
        value_red = data[0]

        # append onto a list
        list_of_red.append(value_red)
        list_of_values.append(value)

        # add to y_sum and move pixel x for 1
        y -= 1
        x_sum += x_increase_final_abs
        if x > total_x or y > total_y:
            break
    
```

<sup>13</sup> smyčka, běží dokud je podmínka platná

# Jaký je výsledek?

- získáme list obsahující jednotlivé průměrné hodnoty pixelů
- a také list obsahující hodnoty jen v červeném spektru
- ty poté dále vložíme do jednotlivých funkcí na výpočet vzdálenosti mezi stínem a mrakem

```
[153, 146, 145, 143,  
157, 192, 221, 228,  
233, 241, 236, 230,  
208, 193, 182, 173,  
165, 155, 153, 155,  
151, 147, 142, 140,  
138, 139, 140, 139,  
138, 138, 130, 119,  
115, 103, 100, 098,  
097, 099, 100, 098,  
099, 113, 117, 123,  
127, 134, 140, 144]
```

# Metoda první: minimum-maximum

- v listu najdeme nejvyšší a nejnižší hodnotu
- zjistíme pozice těchto hodnot v listu
- vypočteme vzdálenost mezi stínem a mrakem

```
def calculate_using_min_max(list_of_values):
    def main():
        # find items in list corresponding to the
        # lowest and highest point
        shadow_low = min(list_of_values)
        cloud_high = max(list_of_values)

        # find of said items in the list (their
        # order)
        shadow_location = list_of_values.index
        (shadow_low)
        cloud_location = list_of_values.index
        (cloud_high)

        # find the difference
        shadow_length = shadow_location -
        cloud_location
        return shadow_length, cloud_high,
        shadow_low, cloud_location,
        shadow_location
```

# Metoda první: minimum-maximum

- když získáme záporný výsledek, stín je před mrakem
- tento pixel můžeme smazat - neovlivní to vzdálenost mezi skutečným stínem a mrakem
- opakujeme dokud nám nevyjde kladný výsledek

```
while True:  
    # in case that shadow is found before a  
    # cloud in the line, we delete the value as  
    # its false and repeat  
    if shadow_lenght <= 0:  
        list_of_values.remove(cloud_high)  
        shadow_lenght, cloud_high,  
        shadow_low, cloud_location,  
        shadow_location = main()  
    else:  
        break
```

# Metoda druhá: maximum-change

- nejprve data překonvertujeme na změny
- i.e. první derivativ původní funkce
- odečteme od sebe sousedící pixely a dáme do nového listu

```
while True:  
    try:  
        current_data = list_of_values[n]  
        previous_data = list_of_values  
        [n-1]  
        change_in_data =  
        current_data - previous_data  
        if n == 0:  
            pass  
        else:  
            list_of_changes.append  
            (change_in_data)  
        n+=1  
    except:  
        break
```

## Metoda druhá: maximum-change

- poté opět zjistíme umístění těchto hodnot v listu
- podle pozice opět vypočteme vzdálenost mezi stínem a mrakem

```
# self-explanatory
shadow_low = max(list_of_changes)
cloud_high = min(list_of_changes)

shadow_location = list_of_changes.index
(shadow_low)
cloud_location = list_of_changes.index
(cloud_high)

# find difference between the two pixel
lengths
shadow_length = shadow_location -
cloud_location
return shadow_length, cloud_high,
cloud_location
```

# Metoda druhá: maximum-change

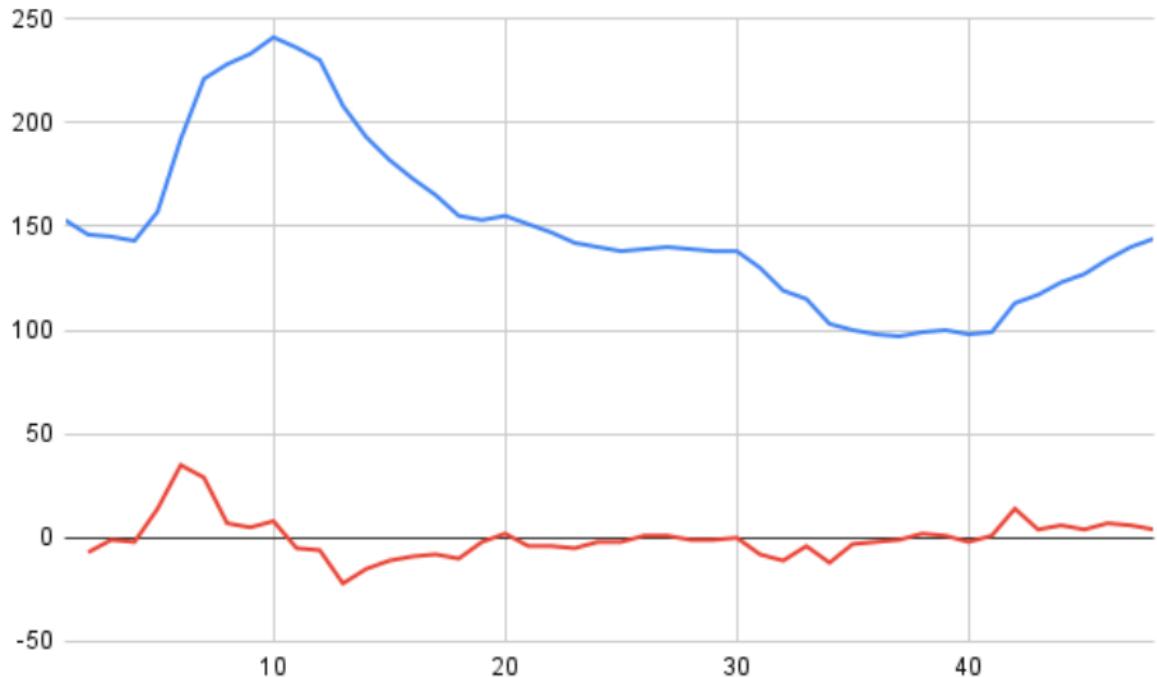
- nakonec opět přidáme ochranu proti záporným výsledkům
- smyčka běží, dokud není výsledek kladný
- začneme v listu hledat až na pozici 10, aby nedošlo k chybě, kdy **nejvyšší nárůst není způsoben stínem ale mrakem**

```
# self-explanatory
shadow_low = max(list_of_changes)
cloud_high = min(list_of_changes)

shadow_location = list_of_changes.index
(shadow_low)
cloud_location = list_of_changes.index
(cloud_high)

# find difference between the two pixel
lengths
shadow_length = shadow_location -
cloud_location
return shadow_length, cloud_high,
cloud_location
```

# Jak vypadá výsledek v grafu?



# Zprůměrování metod

- nyní máme definované všechny potřebné funkce a metody
- využijeme tři výsledky
  - max-change na průměru
  - max-change v červené
  - min-max na průměru
- metody zprůměrujeme
- tím získáme vzdálenost mezi stínem a mrakem
- délku z pixelů převedeme na metry pomocí konstanty

```
# here we calculate the shadow-cloud distance by
# each respective method
shadow_lenght_min_max = calculate_using_min_max
(list_of_values)
shadow_lenght_max_difference =
calculate_using_maximum_change(list_of_values)
shadow_lenght_max_difference_red =
calculate_using_maximum_change(list_of_red)

# we put the methods together
shadow_lenght_final =
(shadow_lenght_max_difference
+shadow_lenght_min_max
+shadow_lenght_max_difference_red)/3

# calculate distance based on a distance in pixels
lenght = int(shadow_lenght_final) * 126.48
```

# Korekce výsledku

- vypočtenou délku je třeba přepočítat z odvěsný na přeponu
- jak je vidět v (animaci), počet pixelů odpovídá delší odvěsně, nikoliv přeponě
- výpočet jsou prosté goniometrické funkce

```
# because original lengths are adjacent
lengths and not hypotenuse we will have
to convert
angle_final_radians = angle_final*(np.pi/
180)
if angle_final <= 45:
    lenght = lenght/np.cos
    (angle_final_radians)
if angle_final > 45:
    lenght = lenght/np.sin
    (angle_final_radians)
```

# Finalizace výsledku

- pomocí již zmíněné rovnice  $h = \tan \alpha \cdot s$  vypočteme výšku mraku
- ještě předtím pomocí třídy sun\_data zjistíme výšku Slunce nad obzorem (i.e. alfa)
- výsledek shadow\_length, který jsme doposud počítali v rovnici odpovídá  $s$

```
# because original lengths are adjacent
lengths and not hypotenuse we will have
to convert
angle_final_radians = angle_final*(np.pi/
180)
if angle_final <= 45:
    lenght = lenght/np.cos
    (angle_final_radians)
if angle_final > 45:
    lenght = lenght/np.sin
    (angle_final_radians)
```

# Nadpis

## Podnadpis

běžný text, struktura stránky, **zvýrazněný text**

- položka odrážkového seznamu na jeden řádek
- položka odrážkového seznamu dlouhá, moc dlouhá (to aby se zalamila), která navíc obsahuje **zvýrazněný text**
  - odrážka druhé úrovně
    - odrážka třetí úrovně
  - **zvýrazněná odrážka druhé úrovně**

### 1. a číslovaná odrážka

1.1 odrážka druhé úrovně se vzorečkem

$$E = mc^2$$

a s odkazem na bibliografickou citaci [1]

# Textové bloky

text nad blokem<sup>6</sup>

Blok

text v bloku

Blok s příkladem

text v bloku

Zvýrazněný blok

text v bloku

---

<sup>6</sup>text poznámky s <https://adresou.cz>

# Tabulky

Jméno	Příjmení	Věk
Albert	Einstein	144
Marie	Curie	155
Thomas	Edison	176

Tabulka: Velcí vědci 19. století

# Bibliografie I

- [1] Albert Einstein. „Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig?“ In: *Annalen der Physik* 323.13 (1905), s. 639–641.
- [2] Edje Electronics. *How to Train TensorFlow Lite Object Detection Models Using Google Colab*. Youtube. 2023. URL: <https://www.youtube.com/watch?v=XZ7FYAMCc4M>.
- [3] Jacob Høxbroe Jeppesen et al. „A cloud detection algorithm for satellite imagery based on deep learning“. In: *Remote sensing of environment* 229 (2019), s. 247–259.

Děkuji vám za pozornost!



ZÁPADOČESKÁ  
UNIVERZITA  
V PLZNI