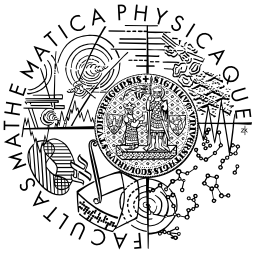


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Ondřej Krsička

Kalibrace meteorologických webových kamer

Katedra softwaru a výuky informatiky

Vedoucí bakalářské práce: Mgr. Martin Mirbauer
Studijní program: Informatika
Studijní obor: Umělá inteligence

Praha 2023

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Děkuji svému vedoucímu Mgr. Martinu Mirbauerovi za vstřícnost a cenné rady při psaní této bakalářské práce. Dále děkuji svým rodičům za podporu při studiu.

Název práce: Kalibrace meteorologických webových kamer

Autor: Ondřej Krsička

Katedra: Katedra softwaru a výuky informatiky

Vedoucí bakalářské práce: Mgr. Martin Mirbauer, Katedra softwaru a výuky informatiky

Abstrakt: Práce se zabývá kalibrací meteorologických webových kamer, která může sloužit k následnému trénování neuronových sítí pro tvorbu realistického osvětlení a generování realistických mraků na obloze do 3D scén. Je srovnána metoda kalibrace podle pozice slunce, podle vzhledu oblohy s využitím analytického modelu oblohy a pomocí neuronových sítí. První dvě metody jsou implementovány v Pythonu. Do metody kalibrace podle vzhledu oblohy je také zakomponován sofistikovanější model oblohy a srovnán s původním modelem oblohy a s kalibrací podle pozice slunce. Kalibrace podle pozice slunce je nejúspěšnější, kalibrace původním modelem o něco méně. Sofistikovanější model oblohy způsobem použitým v práci kalibraci nezlepšuje a jsou navržena možná zlepšení.

Klíčová slova: kalibrace, webová kamera, model oblohy

Title: Weather Webcam Calibration

Author: Ondřej Krsička

Department: Department of Software and Computer Science Education

Supervisor: Mgr. Martin Mirbauer, Department of Software and Computer Science Education

Abstract: The work deals with the calibration of meteorological web cameras, which can be then used for training of neural networks for the creation of realistic lighting and the generation of realistic clouds in the sky into 3D scenes. A method of calibration using the sun position, a method using the sky appearance and a simple analytical sky model and a method based on neural networks are compared. The first two methods are implemented in Python. A more sophisticated sky model is incorporated into the sky appearance calibration method and compared to the original sky model and the sun position calibration. Sun position calibration is the most successful, calibration with the original sky model a little less. The more sophisticated sky model doesn't improve the calibration the way it is used and further improvements are suggested.

Keywords: calibration, webcam, sky model

Obsah

Úvod

Meteorologické webové kamery představují významný zdroj informací o počasí, oblačnosti a dalších atmosférických jevech. Kromě přímého využití v meteorologii mohou tyto kamery sloužit i jako zdroj trénovacích dat pro neuronové sítě generující realistické mraky na obloze. Sběr velkého množství obrazových dat z webových meteorologických kamer je pro výzkumníky neuronových sítí méně náročný než focení oblohy ručně vlastními silami. Tyto neuronové sítě mají uplatnění například ve filmovém průmyslu, videohrách či simulacích pro vytvoření realistického pozadí a osvětlení scény. Pro dosažení vysoké úrovně realismu je důležitá správná kalibrace těchto webových kamer, což zahrnuje určení parametrů kamery, jako jsou azimut, zenit, natočení a zorné pole. Tyto údaje však nemusejí být známé, proto existují metody pro jejich aproximaci.

Cílem této bakalářské práce je porovnat stávající přístupy k automatické kalibraci kamer, konkrétně metody Lalonde a kol. (2010) a Hold-Geoffroy a kol. (2022). Následně je zamýšleno vylepšit první zmíněnou metodu integrací modelu oblohy Prague Sky Model (Wilkie a kol., 2021).

Pro validaci navrženého řešení bude využit dataset webových kamer od Českého hydrometeorologického ústavu (ČHMÚ), který poskytuje obrazová data z 98 webových kamer.

V první kapitole si představíme základní teorii nutnou k porozumění problematiky, jako jsou různé systémy souřadnic a představení modelů oblohy. Ve druhé kapitole se zaměříme matematickou formulaci optimalizačního problému kalibrace kamery různými přístupy. Ve třetí kapitole jsou představena data využitá experimentům. Čtvrtá kapitola pak povrchově popisuje návrh implementace a uživatelské rozhraní programu. Na ni navazuje poslední pátá kapitola, ve které jsou představeny experimenty využívající implementaci popsanou ve čtvrté kapitole, jak pracuje s daty představenými ve třetí kapitole.

1. Teoretický základ

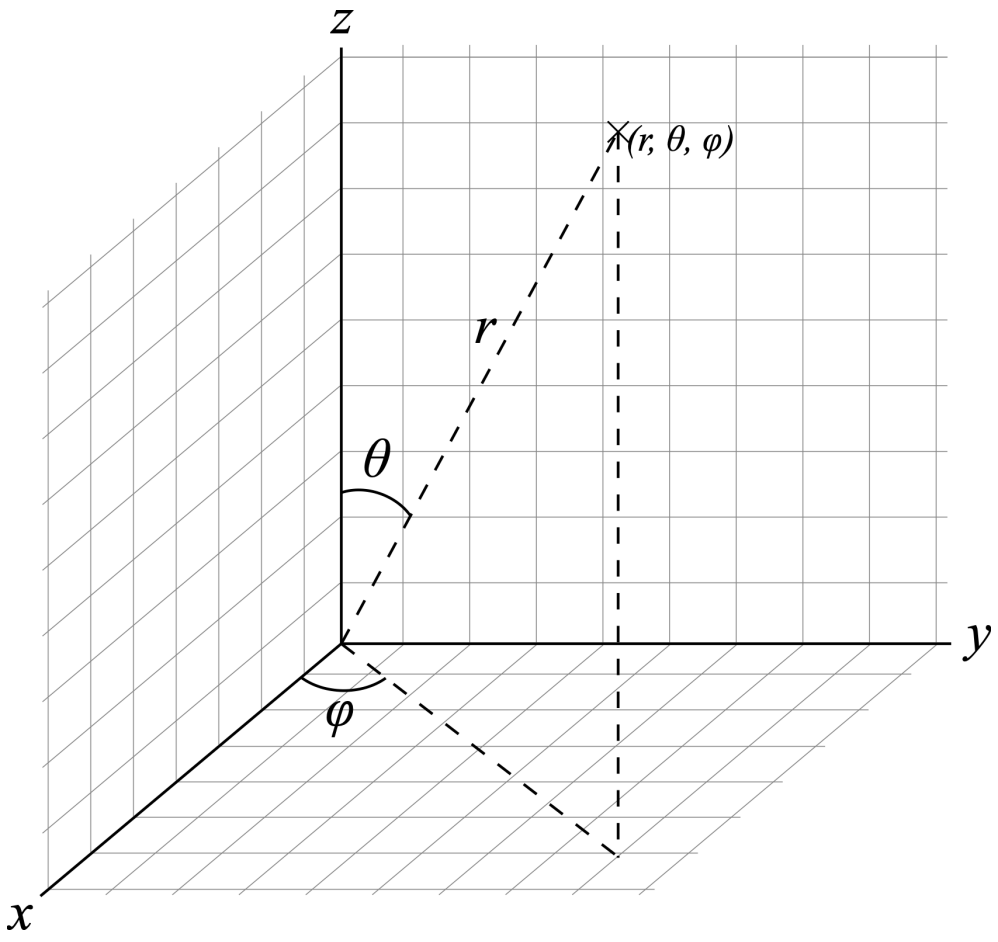
1.1 Sférická soustava souřadnic

Bod na obloze budeme určovat ve sférické soustavě souřadnic. Ta umožňuje bod na obloze jednoznačně určit jeho azimutem ϕ a zenitem θ.

Definice 1 (Azimut)**.** *Azimut ϕ je úhel na vodorovné rovině, který je svírán se severním směrem. Hodnoty azimutu ϕ se pohybují v rozmezí $0°$ až $360°$.*

Definice 2 (Zenit)**.** *Zenit θ je úhel vertikálního směru, měřený od bodu přímo nad kamerou (zenit) směrem dolů k horizontu. Hodnoty zenitu θ se pohybují v rozmezí $0°$ až $90°$. Často používaný pojem je i elevation $= 90° - zenith$, v této práci ale pro přehlednost nebude využíván.*

Orientaci kamery v terénu pak definujeme pomocí úhlů (θ_c, ϕ_c) určujících bod na obloze, který je uprostřed snímku.



Obrázek 1.1: Sférická soustava souřadnic Wikipedia (2023)

1.2 Parametry kamery

Rozlišujeme vnější a vnitřní parametry kamery.

1.2.1 Vnitřní parametry kamery

Vnitřní parametry kamery jsou např. ohnisková vzdálenost f_c, vertikální a horizontální zorný úhel, velikost clony, rychlost závěrky nebo světelnost objektivu. Nejvýznamější vnitřní parametr objektivu je ohnisková vzdálenost f_c, pomocí které lze určit zorné pole kamery.

Definice 3 (Ohnisková vzdálenost)**.** *Ohnisková vzdálenost f_c [mm] je vzdálenost mezi optickým středem objektivu a rovinou snímače (čipu) při zaostření na nekonečno. (Hoško, 2014)*

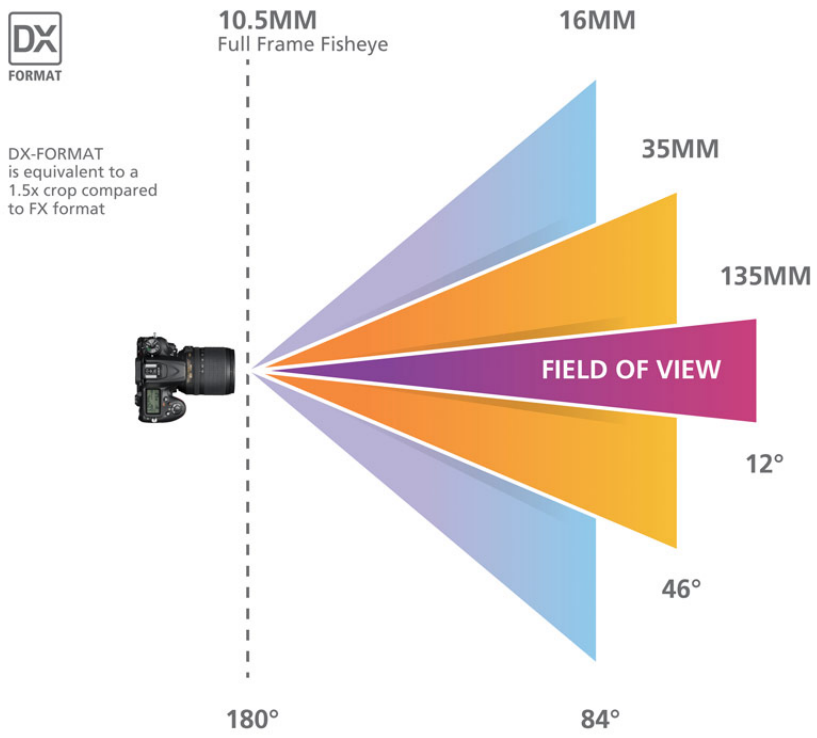
Ohnisková vzdálenost se mění, když upravujeme optický zoom. Větší zoom znamená větší ohniskovou vzdálenost a užší zorný úhel, jak je znázorněno na obrázku 1.2. My budeme ohniskovou vzdálenost modelovat v pixelech, protože neznáme velikost senzoru webových kamer, ale známe rozlišení jejich snímků.

Lemma 1. *Pro objektiv s ohniskovou vzdáleností f_c [px] a snímací čip velilosti (W, H) [px] je zorný úhel ve vodorovném směru ω_h dán vztahem*

$$\omega_h = 2 \arctan\left(\frac{W}{2f}\right). \tag{1.1}$$

Důkaz. Vztah je dán goniometrií pravoúhlého trojúhelníku, jehož odvěsny jsou f a $W/2$.

□



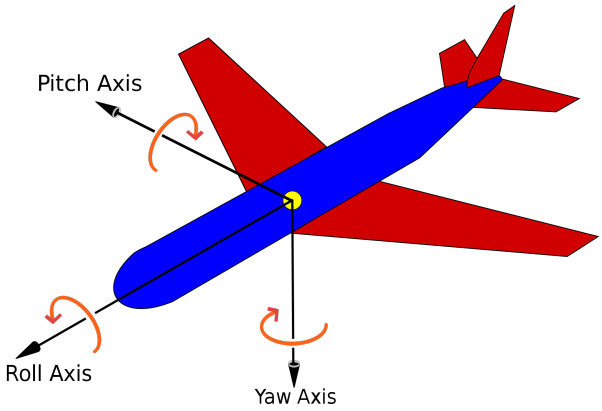
Obrázek 1.2: Vliv ohniskové vzdálenosti na zorný úhel Nikon (2023)

1.2.2 Vnější parametry kamery

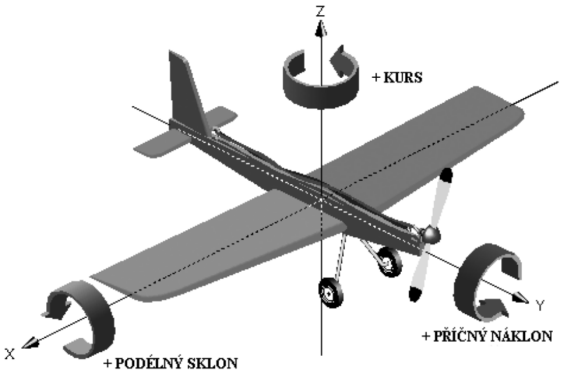
Mezi vnější parametry kamery patří její orientace (zenit θ_c, azimut ϕ_c). Alternativou, jak popsat směřování kamery, je s pomocí úhlů kolem osy x, y a z:

- θ' - úhel kolem osy x, označovaný jako podélný sklon nebo anglicky pitch, $\theta' = \pi/2 - \theta_c$
- ψ' - úhel kolem osy y, označovaný jako příčný náklon nebo anglicky roll
- ϕ' - úhel kolem osy z, označovaný jako azimut, kurs nebo anglicky yaw

Názorně jsou tyto úhly znázorněny na modelech letadla na obrázcích 1.3 a 1.4.



Obrázek 1.3: Polohové úhly Wikipedia (2023)



Obrázek 1.4: Polohové úhly (Famfulík, 2008)

1.3 Nalezení bodu na obloze

Abychom mohli modelovat snímek oblohy kamerou s parametry (f_c, θ_c, ϕ_c), musíme převést pozici pixelu na snímku na pozici bodu na obloze. Předpokládáme, že horizont je rovnoběžný se spodní hranou snímku.

Lemma 2. *(Lalonde a kol., 2010, Appendix B) Buď ϕ_c azimut kamery, θ_c zenit kamery, f_c [px] ohnisková vzdálenost kamery, (W, H) [px] rozlišení snímku, (x_p, y_p) pozice pixelu (horizontální a vertikální vzdálenost od levého horního rohu snímku). Nechť $u_p = x_p - W/2$, $v_p = H/2 - y_p$. Pak*

$$\theta_p = \arccos\left(\frac{v_p \cdot \sin(\theta_c) + f_c \cdot \cos(\theta_c)}{\sqrt{f_c^2 + u_p^2 + v_p^2}}\right), \tag{1.2}$$

$$\phi_p = \arctan\left(\frac{f_c \cdot \sin(\phi_c) \cdot \sin(\theta_c) - u_p \cdot \cos(\theta_c) - v_p \cdot \sin(\theta_c) \cdot \cos(\phi_c)}{f_c \cdot \cos(\theta_c) \cdot \sin(\phi_c) + u_p \cdot \sin(\theta_c) - v_p \cdot \cos(\theta_c) \cdot \cos(\phi_c)}\right) \tag{1.3}$$

Důkaz. Jednotlivé kroky důkazu jsou podrobně popsány v práci Lalonde a kol. (2010, Appendix B).

□

Lemma 3. *(Lalonde a kol., 2010, strana 16) Nechť (ϕ_c, θ_c) je směr kamery, (ϕ_s, θ_s) je pozice slunce na obloze, $f_c[px]$ ohnisková vzdálenost kamery, (W, H) rozlišení snímku, (x_p, y_p) pozice pixelu (horizontální a vertikální vzdálenost od levého horního rohu snímku). Nechť $u_p = x_p - W/2$, $v_p = H/2 - y_p$. Pak úhel γ_p mezi sluncem (ϕ_s, θ_s) a bodem na obloze (ϕ_c, θ_c) je dán vztahem*

$$\gamma_p = \arccos\left(\cos(\phi_p) \cdot \cos(\phi_s) + \sin(\phi_p) \cdot \sin(\phi_s) \cdot \cos(\theta_p - \theta_s)\right) \tag{1.4}$$

Všechny úhly relevantní pro kalibraci kamery jsou znázorněny v obrázku 1.5

1.4 Radiometrie, fotometrie a spektrometrie

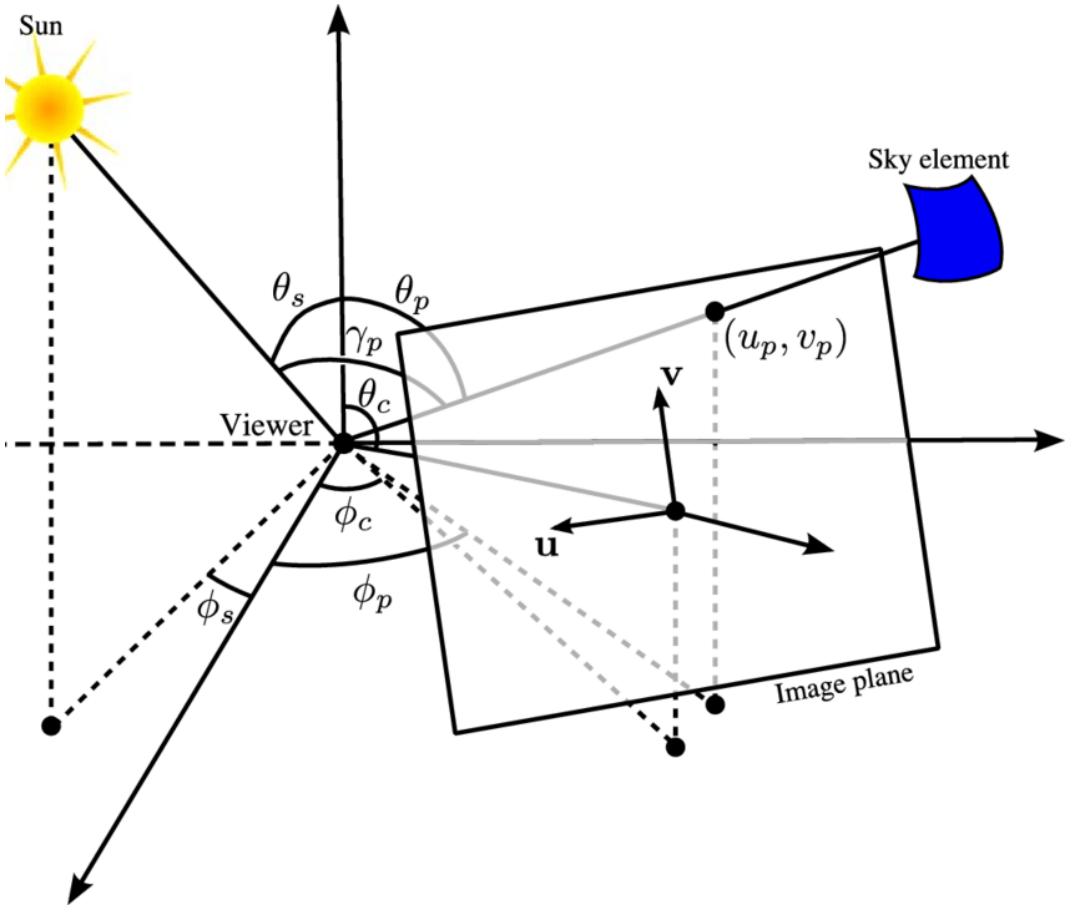
Radiometrie, fotometrie a spektrometrie jsou vědní disciplíny, které se zabývají měřením a analýzou světla a elektromagnetického záření. Radiometrie se zaměřuje na kvantitativní měření energie elektromagnetického záření. Fotometrie se na druhou stranu zaměřuje na měření světla ve vztahu k lidskému vnímání. Spektrometrie zkoumá vlastnosti světla a elektromagnetického záření v různých vlnových délkách. Vzhledem k tomu, že lidské oko má různou citlivost na různé vlnové délky světla, fotometrická měření jsou vážena podle vnímání lidského oka. Citlivost oka na různé vlnové délky je znázorněna v obrázku 1.6.

Nás nejvíce bude zajímat vztah mezi radiometrickou veličinou zář a fotometrickou veličinou jas.

Definice 4 (Spektrální odezva)**.** *Spektrální odezva $V(\lambda)$ je funkce, která popisuje citlivost lidského oka na různé vlnové délky světla.*

Definice 5 (Spektrální zář a jas)**.** *Jas $L[cd/m^2]$ je zář L_e vážená spektrální odezvou lidského oka $V(\lambda)$. Platí:*

$$L(\lambda) = L_e(\lambda) \cdot V(\lambda) \tag{1.5}$$



Obrázek 1.5: Pozice slunce (θ_s, γ_s), orientace kamery (θ_c, γ_c), pozice bodu na obloze (θ_p, γ_p), úhel bodu na obloze se sluncem γ_p, pozice bodu na snímku (u_p, v_p) (Lalonde a kol., 2010, strana 15)

1.5 CIE XYZ

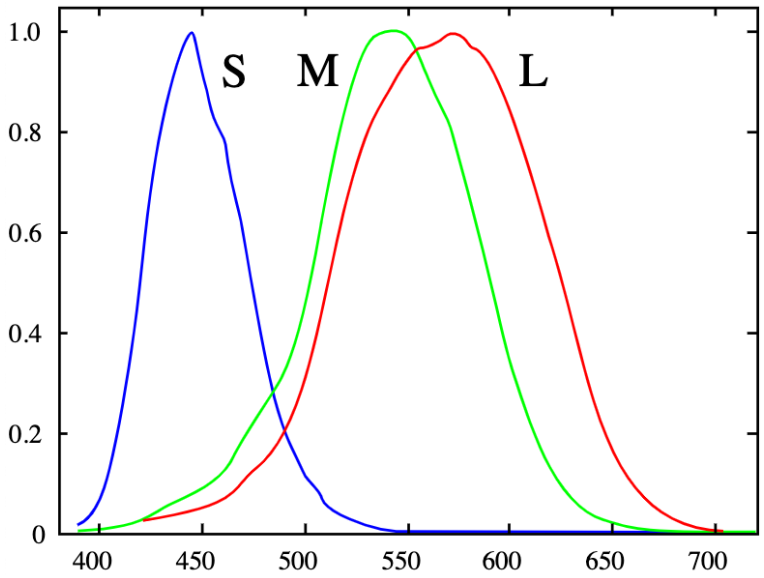
CIE XYZ je matematický model, který se používá k popisu všech barev, které jsou viditelné lidským okem. Osa Y značí jas a nabývá hodnot od 0 (minimální jas) do 1 (maximální jas). Rovina XZ udává chromaticitu, tedy barvu. Pro jednoduchost bude dále v práci využívána pouze složka Y, proto budou snímky generované modely oblohy pouze černobílé (1.7.1).

1.6 Perezův model oblohy

Modely oblohy mají za cíl určit intenzitu světla (celkovou nebo pro danou vlnovou délku nebo barevný kanál) přicházejícího z daného směru z jasné oblohy bez oblačnosti při zvolených parametrech atmosféry a pozice slunce. Tento analytický model oblohy byl poprvé představen v článku Perez a kol. (1993). Ukázka snímku vytvořeného Perezovým modelem je na obrázku 1.7.1 nahoře.

Věta 4. *(Perez a kol., 1993) Relativní jas bodu na obloze vůči jasu zenitu je dán vztahem:*

$$l_p = (1 + a \cdot \exp(b/\cos(\theta))) \cdot (1 + c \cdot \exp(d \cdot \gamma) + e \cdot \cos^2(\gamma)) \tag{1.6}$$



Obrázek 1.6: Normalizovaná spektrální odezva tří typů čípků v lidském oku (Wikipedia, 2023)

Kde (a, b, c, d, e) *jsou koeficienty modelu, které pro jasnou oblohu nabývají hodnot* $(-1, -0.32, 10, -3, 0.45)$.

Věta 5. *(Lalonde a kol., 2010, model oblohy nezávislý na azimutu) Pro* $\gamma_p > 100°$ *lze Perezův model zjednodušit na*

$$l'_p = 1 + a \cdot \exp(b/\cos(\theta)) \tag{1.7}$$

Výhoda Perezova modelu je jeho jednoduchost. Je vhodný pro simulace, ve kterých se preferuje výpočetní výkon nad přesností. Jeho nevýhodou je to, že nedokáže zohledňovat více atmosférických vlivů a že jeho parametry a, b, c, d, e je nutné nalézt v tabulkách nebo nastavit pomocí empirických měření.

1.7 Pražský model oblohy

Pražský model oblohy (Wilkie a kol., 2021) byl vyvinut na Matematicko-fyzikální fakultě Univerzity Karlovy. Oproti Perezovu modelu nepočítá relativní jas (fotometrická veličina), ale zář (radiometrická veličina). Existuje však i verze, která radiometrické veličiny aproximací převádí do barevného prostoru XYZ. Není to analytický model, ale přistupuje k datům vygenerovaným fyzikálním simulátorem atmo_sim postaveném na rendereru ART (Advanced Rendering Toolkit), které jsou komprimovány do souboru o velikosti cca 2GB. Díky tomu je pražský model přesnější a obecnější než Perezův model a zároveň je rychlejší, než fyzikální simulace. Pražský model je ale náročnější na výpočetní i paměťovou kapacitu než analytický Perezův model. Ukázka snímku vytvořeného Pražským modelem oblohy je v obrázku 1.7.1 dole.

1.7.1 Matematický popis

Pražský model oblohy zohledňuje následující proměnné:

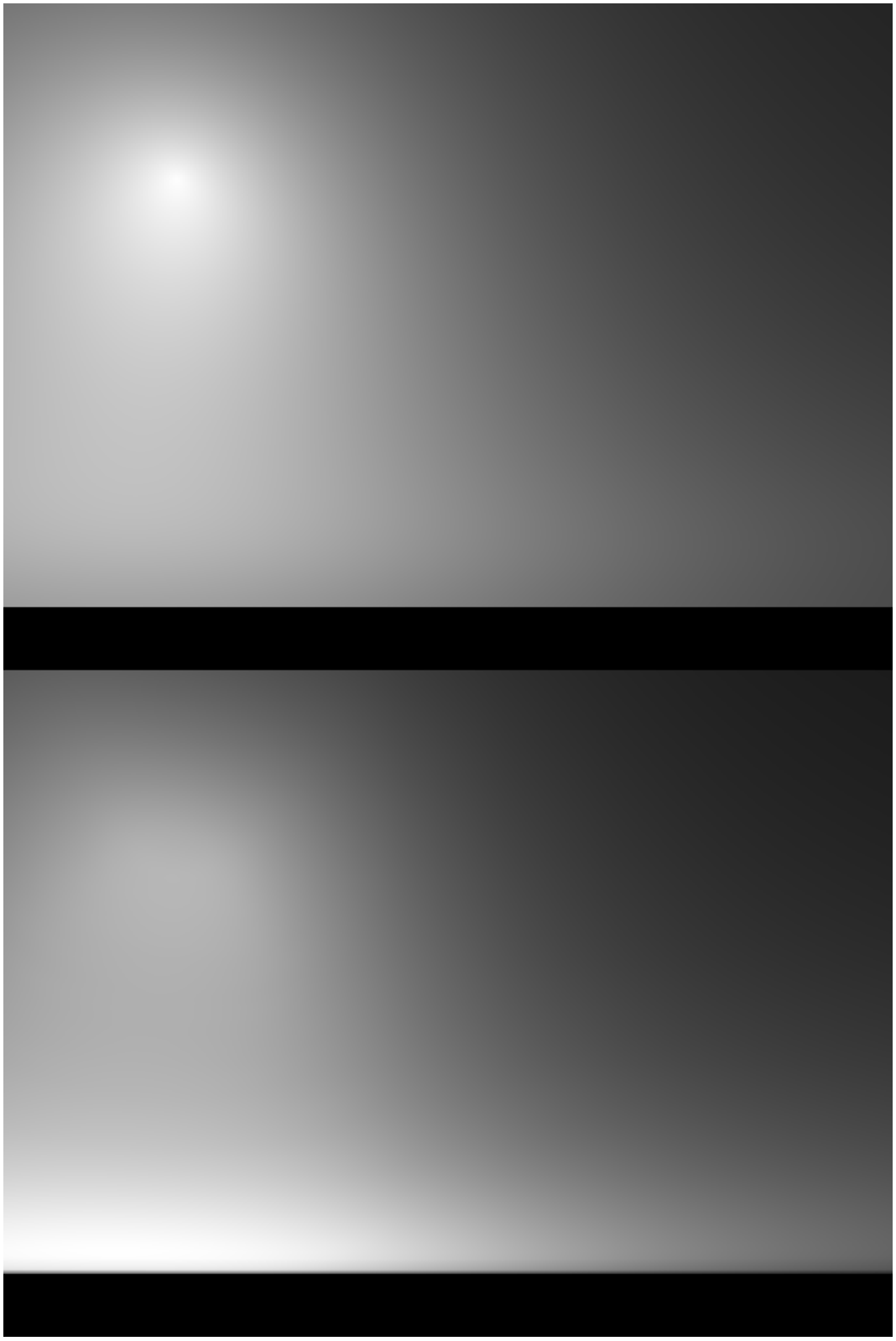
- θ_p: zenit pozorovaného bodu
- γ_p: úhel mezi pozorovaným bodem a sluncem
- ϕ_s: azimut slunce
- θ_s: zenit slunce
- *viditelnost* $[km]$ (za jasného dne zhruba 100 km)
- *albedo*: míra odrazivosti země, v rozsahu od 0 do 1. (Zasněžená krajina má vyšší albedo než travnatá plocha)
- vlnová délka [nm]: pro radiometrickou verzi modelu udává, ke které vlnové délce náleží vrácená spektrální zář.
- kanál: pro upravenou verzi modelu udává, jestli má model vrátit hodnotu X, Y nebo Z. Pro modelování jasu stačí souřadnice Y.

a vrací spektrální zář $L_e[W \cdot sr^{-1} \cdot m^{-2} \cdot nm^{-1}]$ nebo hodnotu v barevném prostoru XYZ.

V práci je využívána neoficiální verze Pražského modelu, která vrací hodnotu CIE XYZ. Díky tomu program nemusí hodnotu jasu integrovat přes vlnové délky a běží tak rychleji. Je také využita verze modelu *Ground*, která předpokládá, že pozorovatel (tedy kamera) se nachází v nadmořské výšce 0 m.n.m.. Úplná verze modelu, která bere jako parametr i nadmořskou výšku, je ještě náročnější a větší než verze *Ground* využitá v této práci.

1.7.2 Využití

Pražský model je použit například ve fotorealistickém renderovacím softwaru Corona Renderer od společnosti Chaos Czech a.s. (Chaos Czech a.s., 2023). Jeho hlavní oblasti využití zahrnují architektonické vizualizace, návrh interiéru, filmový a televizní průmysl, reklamu a 3D vizualizace a animace. Corona Renderer je kompatibilní s programy jako Autodesk 3ds Max a Cinema 4D, což usnadňuje jeho integraci do stávajících pracovních postupů.



Obrázek 1.7: Nahoře ukázka snímku z Perezova modelu oblohy a dole se stejnými parametry z Pražského modelu oblohy.

2. Kalibrace kamery

Kalibrace kamery znamená nalezení jejích vnitřních a vnějších parametrů. Má využití v počítačovém vidění a je nutným krokem pro skládání obrazu z více snímků nebo ve fotogrammetrii, která se zabývá automatickým vytvářením 3D modelů podle reálné předlohy.

Kalibrace kamery se dá provádět s nebo bez pomoci snímků, na nichž se nachází známé kalibrační objekty. V této kapitole se zaměříme na srovnání dvou odlišných metod kalibrace kamery, které nevyžadují kalibrační objekty, a to na Lalonde a kol. (2010) a Hold-Geoffroy a kol. (2022).

2.1 Kalibrace kamery pomocí modelu oblohy

Tuto metodu představil Lalonde a kol. (2010). Předpokládá, že kamera je namířena na oblohu a že se v průběhu času nemění její pozice ani orientace. Dále předpokládá, že kamera je umístěná vodorovně – tedy že horizont je na snímku rovnoběžný se spodní hranou snímku. Vyžaduje z každé kamery několik snímků v průběhu několika dní v různých denních hodinách.

Je založená na optimalizaci čtverce rozdílu obrazu skutečné oblohy a oblohy vygenerované modelem. K nalezení optimálních parametrů modelu oblohy využívá nelineární metodu nejmenších čtverců (non-linear least squares). Algoritmus dokáže určit zenit θ_c, azimut ϕ_c a ohniskovou vzdálenost f_c. Program ke kalibraci vyžaduje sémantickou segmentaci oblohy, aby program nesrovnával krajinu s modelem oblohy. J. F. Lalonde a spol. tuto segmentaci prováděli ručně. K určení f_c a θ_c je využit dataset $\mathcal{I}$, který obsahuje snímky čisté oblohy obsahující pouze vertikální gradient na obloze a k určení ϕ_c využívá dataset $\mathcal{J}_c$, který obsahuje snímky čisté oblohy, na kterých je viditelný i horizontální gradient vzniklý vlivem slunce.

Značení

- $\mathcal{D}_c$ – množina snímků oblohy kamerou c s neznámými parametry (θ_c, ϕ_c, f)
- $y_p^{(i)}$ – jas na pozici (u_p, v_p) v obrázku i
- $\mathcal{P}_c = \{p | segmentace(y_p) = obloha\}$ – pixely označení sémantickou segmentací jako obloha
- $v_{min} = min\{v_p | p \in \mathcal{P}_c\}$ – pozice na ose y nejníže položeného pixelu oblohy
- $\theta_s^{(j)}, \phi_s^{(j)}$ – zenit a azimut slunce v čase pořízení snímku j

2.1.1 Automatická tvorba datasetu $\mathcal{I}_c$ a $\mathcal{J}_c$

J. F. Lalonde představil následují metodu pro selekci snímků vhodných ke kalibraci.

Množina $\mathcal{I}_c$ Chceme najít množinu $\mathcal{I}_c \subseteq \mathcal{D}_c$, která obsahuje snímky čisté oblohy bez oblačnosti, se sluncem co nejdále od zorného pole kamery.

Toho docílíme aproximací každého snímku oblohy pomocí

$$\hat{y}_p^{(i)} = \alpha_i (v_p - v_{min})^2 + \beta_i, \tag{2.1}$$

Parametry α_i a β_i určíme metodou nejmenších čtverců:

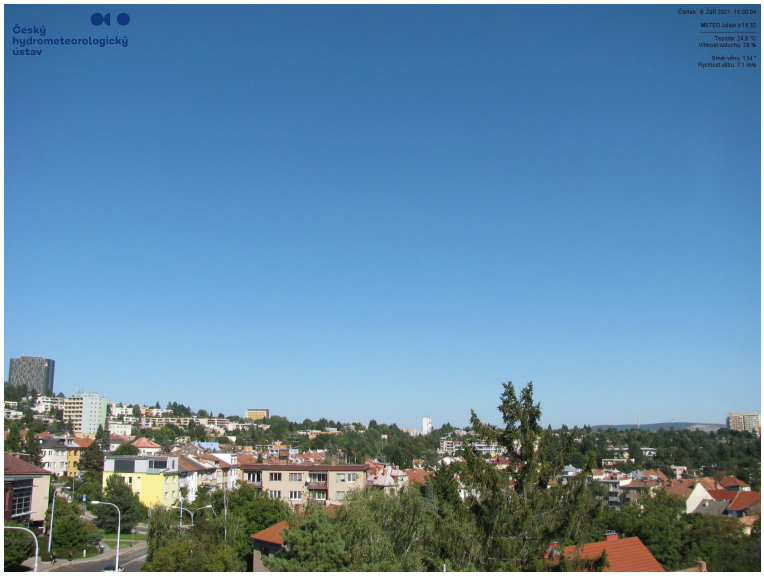
Lemma 6 (Nalezení (α_i, β_i) metodou nejmenších čtverců)**.** *Nechť* $P = |\mathcal{P}_c|$

$$A = \begin{bmatrix} (v_1 - v_{min})^2 & 1 \\ \vdots & \vdots \\ (v_P - v_{min})^2 & 1 \end{bmatrix}, b = \begin{pmatrix} y_1 \\ \vdots \\ y_P \end{pmatrix}.$$

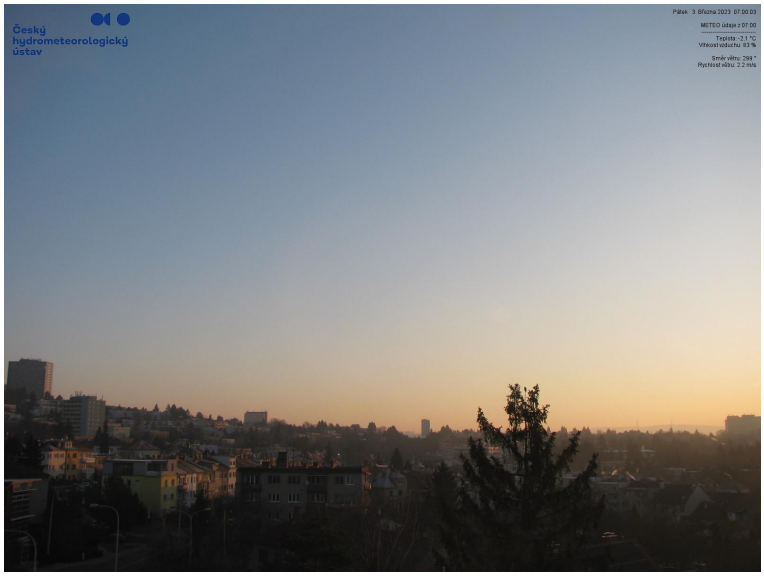
Pak $x = (\alpha_i, \beta_i)^\top = (A^\top A)^{-1} A^\top b$.

Abychom vytvořili $\mathcal{I}_c$, vybereme snímky s $\alpha_i < 0$, poté z nich vybereme 10 % s nejnižším reziduem $||Ax - b||$ a potom vezmeme N s nejnižší $|\alpha_i|$.

Množina $\mathcal{J}_c$ Dataset $\mathcal{J}_c \subseteq \mathcal{D}_c$ sloužící k nalezení zenitu ϕ_c obsahuje snímky jasné oblohy, kde jsou patrné známky toho, že slunce je blízko zorného pole kamery. Není však žádoucí, aby slunce bylo přímo na snímku, protože pak by ve snímku nastaly problémy s expozicí a šumem. Lalonde navrhl $\mathcal{J}_c$ vytvořit tak, že vybere 4 dny s nejvyšším počtem snímků s $\alpha < 0$. Tím by měl získat slunečné dny, protože snímky obsahující mraky méně odpovídají funkci 2.1. Poté ze snímků pořizovaných v průběhu celého dne vybere N snímků s nejmenším vertikálním a horizontálním gradientem, na kterých by měla být nejčistší obloha.



Obrázek 2.1: Příklad snímku z datasetu $\mathcal{I}_c$ (Český hydrometeorologický ústav, 2023)



Obrázek 2.2: Příklad snímku z datasetu $\mathcal{J}_c$ (Český hydrometeorologický ústav, 2023)

2.1.2 Optimalizace

Hledané parametry θ_c, ϕ_c, f_c najdeme nelineární metodou nejmenších čtverců algoritmem Levenberg-Marquardt (LM, Levenberg 1944). Nechť l'_p je výsledek funkce $g'(u_p, v_p, \theta_c, f_c)$ a l_p je výsledek funkce $g(u_p, v_p, \hat{\theta}_c, \phi_c, \hat{f}_c, \theta_s^{(j)}, \phi_s^{(j)})$.

$$\hat{\theta}_c, \hat{f}_c, \hat{k}^{(i)} = \arg\min_{\theta_c, f_c, k^{(i)}} \sum_{i \in \mathcal{I}_c} \sum_{p \in \mathcal{P}_c} \left(y_p^{(i)} - k^{(i)} g'(u_p, v_p, \theta_c, f_c) \right)^2 \tag{2.2}$$

$$\hat{\phi}_c, \hat{k}^{(j)} = \arg\min_{\phi_c, k^{(j)}} \sum_{j \in \mathcal{J}_c} \sum_{p \in \mathcal{P}_c} \left(y_p^{(j)} - k^{(j)} g(u_p, v_p, \hat{\theta}_c, \phi_c, \hat{f}_c, \theta_s^{(j)}, \phi_s^{(j)}) \right)^2 \tag{2.3}$$

Koeficienty $k^{(i)}$ a $k^{(j)}$ v optimalizaci zajišťují to, že výsledek optimalizace nezávisí na pevné hodnotě pixelů jednotlivých snímků a díky tomu zvládne program pracovat i s přeexponovanými nebo podexponovanými snímky.

2.2 Kalibrace kamery podle pozice slunce

Další metoda zmíněná v článku Lalonde a kol. (2010) používá ke kalibraci kamery pozici slunce. Aby fungovala, je nutné jí poskytnout na vstup alespoň 4 obrázky ze stabilní kamery, na kterých lze označit střed slunce a je nutné znát také GPS souřadnice kamery a datum a čas pořízení snímku. Mezi jednotlivými snímky by měla být určitá časová prodleva, aby slunce nebylo pokaždé na stejném místě. Ze znalosti GPS souřadnic a data a času pořízení snímku lze zjistit úhlovou pozici slunce v době pořízení snímku ve sférických souřadnicích. Díky tomu známe 2D pozice slunce na obrázku a 3D pozici slunce v prostoru. Z toho můžeme vypočítat parametry kamery pomocí PnP (Perspective n Point). Způsobů řešení PnP je více, zde J. F. Lalonde využívá nejprve lineární metodu nejmenších čtverců. Její výsledek je poté doupraven pomocí LM algoritmu (Levenberg, 1944), který provádí nelineární metodu nejmenších čtverců.

2.2.1 Matematická formulace

Lemma 7. *(Lalonde a kol., 2010, Appendix A).*

Mějme pozice slunce na obrázku $(u_1, v_1), \ldots, (u_4, v_4)$ *a k nim náležící pozice slunce v prostoru* $(x_1, y_1, z_1), \ldots, (x_4, y_4, z_4)$ *značené* $s_1, \ldots, s_4$*. Nechť*

$$P = \begin{bmatrix} x_1 & y_1 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 & -u_1z_1 \\ 0 & 0 & x_1 & y_1 & z_1 & -v_1x_1 & -v_1y_1 & -v_1z_1 \\ x_2 & y_2 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 & -u_2z_2 \\ 0 & 0 & x_2 & y_2 & z_2 & -v_2x_2 & -v_2y_2 & -v_2z_2 \\ x_3 & y_3 & 0 & 0 & 0 & -u_3x_3 & -u_3y_3 & -u_3z_3 \\ 0 & 0 & x_3 & y_3 & z_3 & -v_3x_3 & -v_3y_3 & -v_3z_3 \\ x_4 & y_4 & 0 & 0 & 0 & -u_4x_4 & -u_4y_4 & -u_4z_4 \\ 0 & 0 & x_4 & y_4 & z_4 & -v_4x_4 & -v_4y_4 & -v_4z_4 \end{bmatrix}$$

,

$$M = \begin{bmatrix} f_c \sin\phi_c & -f_c \cos\phi_c & 0 & 0 \\ -f_c \cos\phi_c \cos\theta_c & -f_c \sin\phi_c \cos\theta_c & f_c \sin\theta_c & 0 \\ \cos\phi_c \sin\theta_c & \sin\phi_c \sin\theta_c & \cos\theta_c & 0 \end{bmatrix}$$

a nechť $m \neq 0$ *minimalizuje* $|Pm|$*. Pak*

$$\begin{aligned} \theta_c &= \arctan\left(\frac{\sqrt{m_{31}^2 + m_{32}^2}}{m_{33}}\right) \\ f_c &= \sqrt{m_{11}^2 + m_{12}^2} \\ \phi_c &= \arctan\left(\frac{m_{11}}{-m_{12}}\right) \end{aligned} \tag{2.4}$$

.

Následně pomocí LM algoritmu hledáme

$$\arg\min_{f_c,\theta_c,\phi_c} \sum_{i=1}^{N} \left(\left(m_1 - u^{(i)}\right) \cdot s^{(i)}\right)^2 + \left(\left(m_2 - v^{(i)}\right) \cdot s^{(i)}\right)^2 \tag{2.5}$$

2.2.2 Výhody a nevýhody metody

Hlavní výhodou oproti předchozí metodě je, že funguje i když je na obloze viditelná oblačnost, což je v České republice velmi častý jev a je velmi náročné najít dny, kdy je obloha zcela bez mraků.

Nevýhodou je, že tato metoda vyžaduje označení pozic slunce, která je sama o sobě netriviální. Ruční anotace je ve velkém měřítku nerealizovatelná. Automatická anotace se bude potýkat s různými jevy, které budou mít negativní vliv na její výsledek. Příkladem mohou být mraky zakrývající slunce, odlesky, mlha, umělé osvětlení, přeexponovanost snímku a další. Tento problém by nemusel nastat u HDR (High Dynamic Range) kamer, které podporují větší rozpětí barev a slunce by tedy vždy mělo řádově vyšší detekovatelnou hodnotu než např. mraky.

Dále touto metodou nelze kalibrovat kamery, které nejsou schopny pořizovat snímky slunce. To jsou v českých podmínkách všechny meteorologické kamery orientované na sever, které nejsou natočené přímo nahoru k zenitu a nepořizují širokoúhlé snímky.

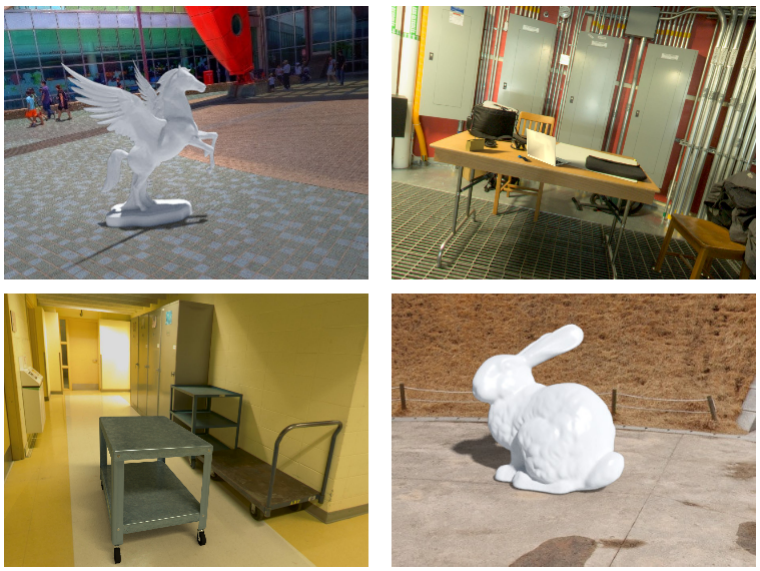
Další nevýhodou je citlivost na chybu v anotaci pozice slunce. I malá nekonzistence, chyba v anotaci může způsobit velkou chybu v kalibraci kamery. To lze řešit např. tím, že program bude zkoušet, zda se vydaří kalibrace s pozicemi slunce posunutými o náhodný šum.

2.3 Kalibrace kamery pomocí hlubokých neuronových sítí

Práce Hold-Geoffroy a kol. (2022) představuje metodu, jak z jediného snímku určit její příčný náklon ψ_c, zenit θ_c, ohniskovou vzdálenost f_c a zkreslení čočky ξ_c. Oproti předchozí metodě má méně specifický vstup a proto i širší použití. Nevyužívá však znalost o pozici slunce v době pořízení snímku, takže nemá jak určit azimut kamery ϕ_c. Spíše než k tvorbě datových podkladů pro trénování neuronových sítí tedy může sloužit ke vkládání 3D objektů do snímků nebo nápravě zkreslení čočky.

Metoda staví na konvoluční neuronové síti s architekturou DenseNet (Huang a kol.) předtrénované na obrazové databázi ImageNet (Deng a kol., 2009). Výstupní vrstva tohoto modelu je potom nahrazena čtyřmi výstupy: náklon ψ, vzdálenost horizontu od středu obrázku v_m, zorný úhel h_θ, a zakřivení ξ. Síť je dotrénována na datasetu 360Cities (360 Cities Holding B.V., 2023). Trénování probíhá pomocí algoritmu Adam (Kingma a Ba, 2014) s využitím ztrátové funkce KL divergence (Kullback a Leibler, 1951).

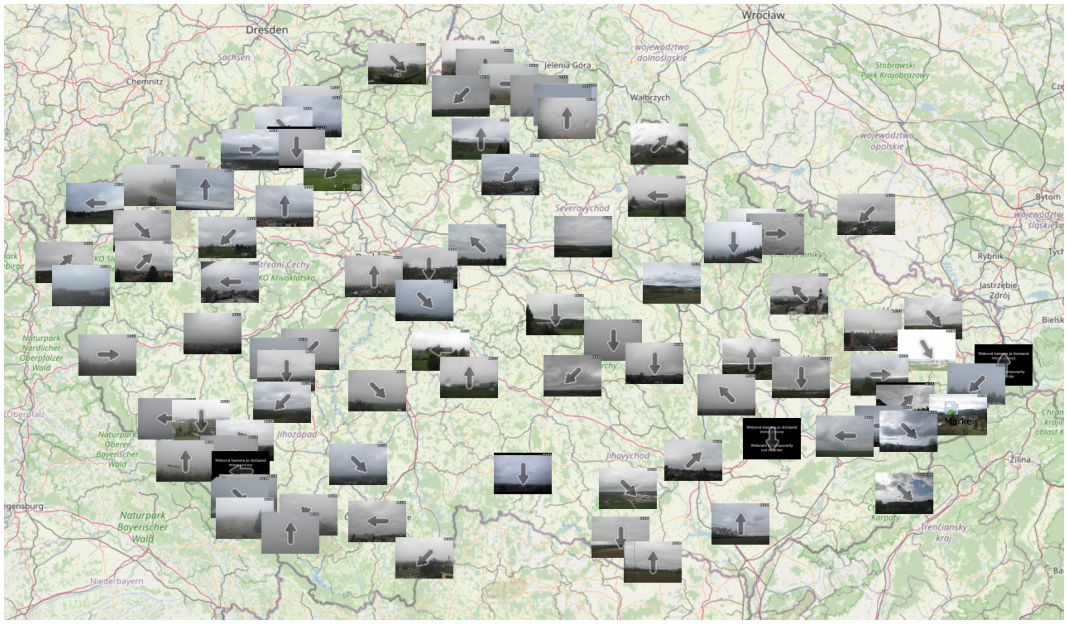
Obrázek 2.3: Automatická oprava zkreslení čočky (Hold-Geoffroy a kol., 2022)



Obrázek 2.4: Realistické vkládání 3D objektů do snímků (Hold-Geoffroy a kol., 2022)

3. Data využitá pro experimenty

Práce využívá obrazová data z veřejně dostupných webových meteorologických kamer Českého hydrometeorologického ústavu (ČHMÚ).



Obrázek 3.1: Mapa webových kamer. Český hydrometeorologický ústav (2023)

Ke srovnání bylo vybráno 10 kamer, na nichž bylo možné provést kalibraci pomocí obou metod, tedy jak podle pozice slunce, tak podle gradientu na obloze.

Všechny použité snímky oblohy v datasetu $\mathcal{I}$ byly pořízeny dne 31.5.2023 a je z nich ručně vybráno ke každé kameře několik desítek snímků s nejčistší oblohou. Bylo tak dosaženo lepších dat oproti automatickému hledání, které navrhuje Lalonde. Rovněž tak byl vybrán dataset $\mathcal{J}$, který i Lalonde z větší části selektoval ručně.

Na obrázku 3.2 je ukázka snímků z datasetu $\mathcal{I}$ každé z 10 kamer a na obrázku 3.3 je stejná ukázka z datasetu $\mathcal{J}$.

3.1 Formát dat

Program podporuje formát dat podobný tomu, který používá Lalonde ve své implementaci v Matlabu a formát více přizpůsobený pro použití nad daty z webových kamer ČHMÚ.

3.1.1 Dataset Arizona

Program podporuje stejný formát dat, jaký používá Lalonde ve své implementaci. Tedy pro jednu kameru máme všechny obrázky z datasetu $\mathcal{I}$ uloženy v jedné složce I a všechny obrázky z datasetu $\mathcal{J}$ v jedné složce J. Ke každému obrázku musí existovat soubor se stejným názvem, ale s příponou *.mat* (např. *1.jpg* a k němu *1.mat*), který v položkách *sunAzimuth* a *sunZenith* obsahuje pozici slunce v

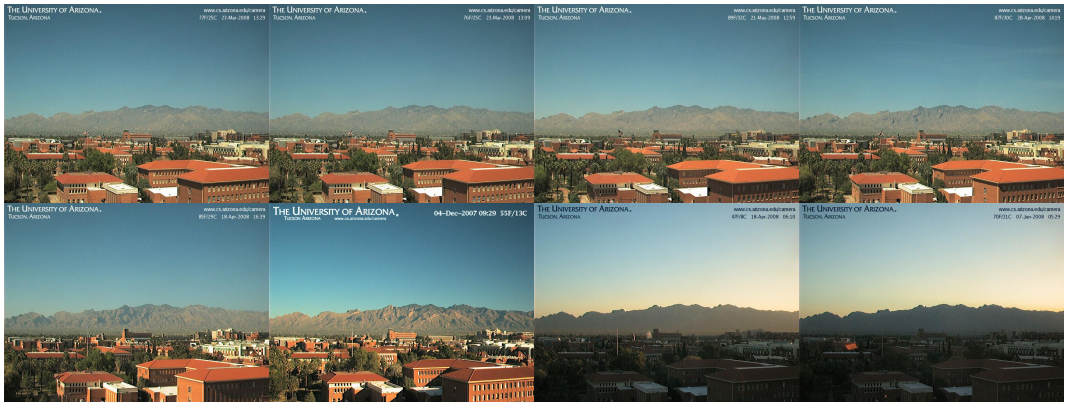
Obrázek 3.2: Každý řádek náleží jedné kameře ČHMÚ a v prvním sloupci obsahuje masku a za ní 5 náhodných snímků z datasetu $\mathcal{I}$.

době pořízení snímku v radiánech. Azimut je zde měřen tak, že východ má azimut -90°, jih -180° a západ -270°. Dále je nutné programu dodat masku oblohy, což je obrázek, na kterém je obloha vyznačena bílou barvou a zbytek černou. Kvůli tomu, že kamera použitá pro Lalondeho demo je z The University of Arizona, je tento formát dat v programu pracovně pojmenován *Arizona*. Ukázka snímků je na obrázku 3.4.

Obrázek 3.3: Každý řádek náleží jedné kameře ČHMÚ a v prvním sloupci obsahuje masku a za ní 5 náhodných snímků z datasetu $\mathcal{J}$.

3.1.2 Dataset ČHMÚ

Druhý podporovaný formát více odpovídá způsobu, jakým jsou uložena data z kamer ČHMÚ na serveru KSVI MFF UK. Počítá se s tím, že se bude kalibrovat více kamer. Místo, datum a čas pořízení snímku jsou zakódované do názvu souboru. Pozice slunce není uložena do souboru, ale vypočítává se za běhu. Na nejvyšší úrovni jsou rozdělené složky I a J, které mají stejnou vnitřní strukturu a poté složka *masks*, která obsahuje masky oblohy. I a J obsahují pro každou kameru jednu složku (*brno*, *temelin*, ...), ke které v *masks* jsou pod patřičnými



Obrázek 3.4: První řádek obsahuje ukázku z množiny $\mathcal{I}$ a druhý řádek z množiny $\mathcal{J}$ z datasetu Arizona.

názvy obsaženy příslušející masky oblohy (*brno.png*, *temelin.png*, ...). Složka pro danou kameru potom obsahuje pro každý den jednu složku (např. *20230531* značí 31.5.2023) která obsahuje všechny snímky z daného dne. Ty jsou pojmenovány podle času pořízení. Např. *I/brno/20230531/1230.jpg* značí snímek pořízený 31.5.2023 v Brně ve 12:30 hodin, který je určen pro použití v datasetu $\mathcal{I}$ pro kalibraci zenitu θ_c a ohniskové vzdálenosti f_c.

GPS souřadnice jsou potom podle názvu kamery nalezeny v souboru JSON, který obsahuje seznam slovníků, v nichž je ke každé kameře uloženo jméno v položce *name* a souřadnice v položkách *lat* a *lon*. Ostatní položky nejsou pro fungování programu nutné. Toto schéma podporuje soubor

`https://www.chmi.cz/files/portal/docs/meteo/kam/webcams.json`,

ze kterého jsou čerpány informace o kamerách ČHMÚ.

3.1.3 Data pro kalibraci z pozice slunce

Kalibrace, která hledá parametry projekce čtyř bodů v prostoru (reálná pozice slunce) na čtyři body na rovině snímku (pozice slunce na snímku) byla prováděna pouze na datech ČHMÚ. Anotace byla provedena ručně a byla uložena do JSON souboru *p4p.json*, který obsahuje tři top-level položky: *locations* obsahující samotné kalibrace, W značící šířku snímku v pixelech a H značící výšku snímku v pixelech. Předpokládá se, že všechny snímky použité pro anotaci mají stejnou výšku a šířku, proto jsou v souboru tyto informace pouze jednou. Je nutné je uvést, protože následně anotace obsahuje absolutní pozice pixelů, které by přestaly mít hodnotu, kdyby se informace o rozlišení snímků ztratily. Položka *locations* obsahuje pro každou kameru slovník se třemi seznamy *filenames* s absolutními cestami k souborům, *xs* se souřadnicemi x a *ys* se souřadnicemi y. Každý z těchto tří seznamů obsahuje právě čtyři položky, protože ke kalibraci jsou nutné čtyři různé pozice slunce na čtyřech různých snímcích. Informace o úhlové pozici slunce na obloze se získá z jména souboru a pozice kamery stejně jako v 3.1.2

Anotace pozic slunce je znázorněna v obrázku 3.5.

Obrázek 3.5: Ruční anotace pozice slunce pro PnP kalibraci kamer ČHMÚ

4. Implementace

Program byl navržen tak, aby byl informovaným uživatelem pohodlně použitelný z příkazové řádky i na jeho vlastních datech, která ovšem musejí být v jednom z formátů popsaném v 3.1. Program umožňuje plnou konfigurovatelnost pomocí CLI argumentů.

4.1 Použité programy, programovací jazyky a knihovny

Program vychází z několika hotových programů. Výchozí Lalondeho implementace napsaná v jazyce Matlab je využita pro ověření správnosti vlastní implementace a je mírně upravena, aby fungovala na datech získaných z ČHMÚ. Dále je využita implementace Pražského modelu oblohy napsaná v jazyce C, pro kterou byl v rámci této práce vytvořen wrapper do Pythonu po vzoru SkyGAN (Mirbauer a kol. 2022)

4.1.1 Matlab

Matlab je komerční programovací jazyk a prostředí pro numerické výpočty a existuje k němu open-source alternativa Octave. V jazyce Matlab je napsána původní implementace kalibrace od Lalondeho, která byla mírně upravena pro ověření správnosti vlastní implementace na datech získaných z ČHMÚ. Matlab nebyl využit pro implementaci vlastního programu, neboť komerční licence komplikuje jeho použití na různých zařízeních a v Docker kontejnerech. Open source alternativa Octave nebyla využita, protože není stoprocentně kompatibilní s Matlabem. Lalondeho implementace webcamCalibration v Octave nebylo možné spustit a pravděpodobně by tak pro spuštění v Octave vyžadovala netriviální úpravy.

4.1.2 Python

Hlavní část programu je napsána v jazyce Python. Byl zvolen kvůli jeho bohatému ekosystému knihoven a jednoduchému použití, což z něj činí vhodný jazyk pro prototypování, strojové učení či zpracování dat.

Pro základní práci s maticemi a poli je využita knihovna *numpy*, která je napsaná v jazyce C a její využití tak činí program výkonnější oproti použití čistě Pythoních struktur. Dále jsou využity funkce pro optimalizaci z knihovny *SciPy*, která rozšiřuje funkcionalitu knihovny *numpy* o další funkce pro vědecké výpočty. Hledání pozice slunce podle souřadnic GPS a času zajišťuje knihovna *pysolar*, která vyžaduje informace o časových zónách, se kterými pracují knihovny *pytz* a *timezonefinder*. Obrázky jsou načítány knihovnou *Pillow*.

4.1.3 C/C++

Díky tomu, že je Python napsaný v C, je možné v Pythonu efektivně využívat funkce napsané v C/C++. Pražský model oblohy napsaný v C by tak bylo možné využít přímo v Pythonu, ale této možnosti nebylo využito. Pro vyšší výkon

byl vytvořen wrapper `sky_image_generator.cpp`, který modeluje více pixelů po dávkách (batch processing) a díky využití paralelizace a nízkoúrovňových datových typů v C++ je výpočet rychlejší než v Pythonu. Je tak docíleno podobného efektu, jaký má knihovna *numpy*.

`sky_image_generator.cpp` je pomocí knihovny *pybind11* zkompilován do Pythoního modulu *sky_image_generator*. Kompilaci provádí kompilátor *g++* s parametry, které mu dodá CLI nástroj *python3-config* obsažený v Pythoním balíčku *python3-dev* a příkaz pro kompilaci tak funguje na různých zařízeních bez nutnosti ručních úprav cesty k Pythonu apod.

4.1.4 Docker a docker-compose

Využitím Dockeru je docíleno toho, že program pokaždé běží v deterministickém předdefinovaném prostředí. Pomocí *Dockerfile* se definuje, jak se má vytvořit Docker image, který lze přirovnat k souboru spustitelné aplikace. Spuštěním Docker image vznikne Docker container, který je běžící instancí Docker image a dá se přirovnat k běžícímu procesu. S jakými parametry se má kontejner spustit, lze definovat jako argumenty v příkazové řádce nebo v souboru *docker-compose.yml*. Konfigurační soubory *Dockerfile* a *docker-compose.yml* tak mimojiné fungují i jako dokumentace, jak program spustit, jaké má závislosti a jaké parametry lze nastavit. Program je otestovaný v Dockeru na operačních systémech Windows, Linux a macOS na architekturách procesoru x86 a arm64. Program nefunguje nativně ve Windows, protože balíček používaný pro sestavení C++ programu *python3-dev* není na Windows k dispozici.

Jediná část programu, která není spouštěna přes Docker, je veškerý kód napsaný v Matlabu. Ten totiž pro nainstalování *Optimization Toolbox* vyžaduje, aby byl uživatel přihlášený svým Mathworks účtem, nebo aby na zařízení byl přítomný licenční soubor vygenerovaný speciálně pro dané zařízení. Generovat nové licence pro každý Docker image jde proti principu Dockeru, který má být co nejvíce deterministický a opakovatelný. Instalace nástroje, který má několik gigabajtů, při každém spuštění by zase způsobila mnohanásobné zpomalení programu a zbytečné zatížení sítě. Řešením by bylo mít jeden long-living kontejner, což ale také jde proti filozofii Dockeru, která prosazuje bezstavové rychle spustitelné kontejnery.

Alternativou je distribuovat program do Matlab Runtime, které pro spuštění programu licenci nevyžaduje. Cílem práce však nebylo vytvořit distribuci Lalondeho programu. Ten je použit pouze pro srovnání výsledků a pro vývoj vlastního programu a proto se funkce v Matlabu spouštějí standardně v prostředí Matlabu.

Izolovat aplikace od ostatních aplikací lze i pomocí virtual machines, ale ty jsou náročnější na výkon, zabírají více místa na disku a spouštějí se pomaleji. Virtuální stroje se tak vyplatí hlavně pro aplikace, které pracují s velmi citlivými daty a vyžadují maximální zabezpečení nebo potřebují komunikovat se speciálním hardwarem. Ani jedno z předchozích není případ této práce.

4.2 Instalace

Ke programu je nutné mít na počítači nainstalovaný Docker, kterým se budou spouštět Linuxové kontejnery. Také je nutné mít nainstalován Git LFS pro stažení

velkých souborů. Po stažení repozitáře se program nainstaluje a spustí následovně:

```
> git clone \
  https://gitlab.mff.cuni.cz/krsickao/prague-sky-model-webcam-calibration
> cd prague-sky-model-webcam-calibration
> docker-compose run bash
```

První spuštění `docker-compose run bash` program nainstaluje (sestaví Docker image) a je proto pomalejší než další běhy, které už instalaci neprovádějí.

4.3 Rozhraní programu

Program je napsán jako CLI nástroj spustitelný v Dockeru. Všechny číselné parametry nebo cesty k souborům, které jsou v programu využity, program získává z CLI argumentů. Díky tomu je možné měnit všechny myslitelné parametry bez zásahu do kódu. Kód neobsahuje žádné výchozí parametry, protože by to do aplikace vneslo chaos a programátor, který by aplikaci chtěl dále rozšiřovat nebo upravovat, by nevěděl, kde jsou tyto výchozí parametry nastaveny a musel by tak procházet celý kód. Tento přístup je ale méně uživatelsky přívětivý, protože každé spuštění vyžaduje mnoho parametrů. To se však dá vyřešit skrze spouštěcí skript, ve kterém jsou výchozí parametry pevně nastaveny.

Výsledek programu je vždy vracen na standardní výstup ve formátu JSON Lines. JSON Lines je datový formát, který na každém řádku obsahuje validní JSON dokument. Výhodou oproti použití seznamu v dokumentu JSON je ten, že JSON Lines je vhodnější pro logování. Do jednoho logu může zapisovat více běhů programu a log je stále validní. Kdybychom se pokoušeli takto zapisovat do dokumentu JSON, byl by dokument validní až po zapsání poslední položky do seznamu bez čárky na konci a po uzavření dokumentu zavírací závorkou. Při použití JSON Lines je výstupní soubour validní stále.

Výstup vždy obsahuje i přesné argumenty, se kterými byl program spuštěn. Díky tomu se tak nestane, že by uživatel ztratil informaci o tom, s jakými parametry kalibraci spustil. To je reálný problém, protože kalibrace více kamer může běžet i v řádu vyšších jednotek hodin, takže ji člověk nechce zbytečně spouštět znovu.

Program pro kalibraci modelem oblohy má následující rozhraní:

```
> docker-compose run bash
[+] Building 0.0s (0/0)
[+] Building 0.0s (0/0)
usage: main.py [-h] [-I I] [-J J] [-m MASK] [-n PX_NUM] [-W W]
               [-H H] [-f0 F0] [-df {date-time,matlab-flat}]
               [-mI {perez,perez-azimuth-independent,prague}]
               [-mJ {perez,prague}] [-min MIN] [-max MAX]
               [-webcams WEBCAMS]

Lalonde's camera calibration of zenith, azimuth and focal length.

options:
  -h, --help            show this help message and exit
  -I I                  Path to folder containing dataset I used for
```

```
                      calibration of focal length and zenith angle.
-J J                  Path to folder containing dataset J used for
                      calibration of azimuth angle.
-m MASK, --mask MASK  Path to sky mask image.
-n PX_NUM, --px-num PX_NUM
                      Number of pixels in each image to be used
                      for calibration.
-W W                  Width of images in dataset. Will be resized
                      to this value.
-H H                  Height of images in dataset. Will be resized
                      to this value.
-f0 F0                Initial guess of focal length.
-df {date-time,matlab-flat}, --dataset-format {date-time,matlab-flat}
                      Format of dataset folder.
-mI {perez,perez-azimuth-independent,prague},
--model-I {perez,perez-azimuth-independent,prague}
                      Sky model for dataset I.
-mJ {perez,prague}, --model-J {perez,prague}
                      Sky model for dataset J.
-min MIN              Pixel values lower than this will be ignored
                      for calibration.
-max MAX              Pixel values higher than this will be
                      ignored for calibration.
-webcams WEBCAMS      Path to JSON file containing webcam
                      positions. Only used when --dataset-
                      format=date-time
```

4.4 Architektura

Program je navržen objektově. Nejvýznamnější část tvoří třídy `ArizonaCalibration`, od které dědí `CHMUCalibration`, potom `CoordinateConvertor`, abstraktní třída `SkyModel`, od které dědí `PerezSkyModel`, `PerezAzimuthIndependentSkyModel` a `PragueSkyModel`. Kalibraci z pozice slunce provádí `PerspectiveCalibrator` a pozice slunce se počítá v `SunPositionCalculator`.

4.4.1 Třídy reprezentující modely oblohy

Všechny modely oblohy vytvářejí pouze jednu hodnotu pro každý pixel a není tak možné s nimi generovat barevné obrázky, pouze černobílé.

Třída `SkyModel` je abstraktní třída, která definuje rozhraní pro model oblohy. Jednotlivé modely oblohy jsou implementovány v třídách `PerezSkyModel`, `PerezAzimuthIndependentSkyModel` a `PragueSkyModel`. Instance třídy SkyModel jsou díky polymorfismu snadno zaměnitelné a modely tak lze snadno srovnávat.

Jedno volání metody `SkyModel.model` bere jako vstupní parametry seznam pozic pixelů, pozici slunce a parametry kamery a vrátí hodnoty, které by měly být na těchto pozicích na obrázku. Je tak možné udělat model celého obrázku, ale také jen vybraných pixelů v obrázku, což pro kalibraci stačí a zrychlí to její výpočet.

Třídy `PerezSkyModel` a `PerezAzimuthIndependentSkyModel` jsou implementovány čistě v Pythonu a numpy. `PragueSkyModel` místo numpy používá modul `sky_image_generator` napsaný v C++ a vytvořený pomocí knihovny pybind11.

4.4.2 Třída CoordinateConvertor

Tato třída slouží jako jediný zdroj pravdy (single source of truth) pro převod souřadnic. Veškeré převody mezi souřadnicovými systémy jsou implementovány v této třídě a změny v souřadnicových systémech jsou tak snadno propagovatelné do všech částí programu. Pomocí této třídy lze převádět mezi všemi následujícími souřadnicovými systémy:

- (x,y): pozice pixelu na obrázku, kde x je vodorovná souřadnice a y svislá souřadnice. Počátek se nachází v levém horním rohu obrázku a všechny pixely mají nezáporné celočíselné souřadnice. Maximální hodnota je vpravo dole a je rovna $(W-1, H-1)$, kde (W, H) jsou šířka a výška obrázku.
- (u, v): pozice pixelu na obrázku, kde u je vodorovná souřadnice a v svislá souřadnice. Počátek se ale nachází uprostřed obrázku a souřadnice v oproti souřadnici y v předchozím bodě roste v opačném směru. Hodnoty jsou v rozmezí $(-W/2, -H/2)$ a $(W/2-1, H/2-1)$
- (ϕ, θ, γ): sférické souřadnice, kde ϕ značí azimut a θ zenit bodu na obloze. γ je úhel mezi směrem (ϕ, θ) a směrem slunce a je nutná pro fungování modelů oblohy.
- (x, y, z): Směrový vektor bodu na obloze. Jsou potřeba pouze kvůli mezivýpočtům a nejsou využívány v žádné jiné části programu. Díky tomu je pro převod ze sférických do kartézských souřadnic možné použít libovolný rádius, aniž by to ovlivnilo výsledek. V programu je nastaven na hodnotu *1*.

K převodům mezi 2D a 3D souřadnicemi je nutné znát parametry kamery, jako je rozlišení, ohnisková vzdálenost a orientace kamery. Rozlišení kamery třída získává v konstruktoru a ostatní parametry jako argumenty jednotlivých metod.

4.4.3 Třídy ArizonaCalibration a CHMUCalibration

Třída ArizonaCalibration implementuje kalibraci kamery pomocí Perezova modelu. Co nejvíce se snaží napodobit Lalondeho implementaci v Matlabu a předpokládá stejný tvar dat, jaký používá Lalonde v `demoSkyCalib.m`. Je zde využita nelineární optimalizace pomocí Levenberg-Marquardtova algoritmu, která využívá obecný model oblohy a je tedy snadné mezi modely přepínat. Výchozí hodnota ohniskové vzdálenosti je získána z parametru a výchozí hodnota zenitu je nastavena tak, aby na snímku rovina horizontu protínala nejníže položený bod oblohy. Azimut je potom kalibrován čtyřikrát s výchozími hodnotami 0°, 90°, 180°a 270°. Následně je z těchto čtyř výsledků vybrán ten výsledek, který má nejnižší ztrátovou funkci.

Třída je navržena tak, aby načítání dat bylo oddělené od logiky kalibrace a vytvořením dědičné třídy je tak možné načítat i data v jiném tvaru.

To dělá třída `CHMUCalibration`, která dědí z `ArizonaCalibration` a přetěžuje pouze metody, které slouží k získávání obrazových dat a dat o pozici slunce. Rozdíl je v tom, že `ArizonaCalibration` načítá informace o pozici slunce ze souborů `.mat`, zatímco `CHMUCalibration` je vypočítává pomocí třídy `SunPositionCalculator`.

5. Experimenty

V této kapitole jsou představeny provedené experimenty a jejich výsledky a vyhodnocení. Jsou zde srovnány výsledky kalibrace pomocí různých modelů oblohy z různě velkého množství obrázků a s různými parametry. Výsledky těchto experimentů jsou potom srovnány s výsledky kalibrace pomocí pozice slunce a s orientací kamer ČHMÚ kvantizovanou na 8 světových stran.

Pro snadnější interpretaci výsledků je místo ohniskové vzdálenosti v pixelech použit horizontální zorný úhel FoV ve stupních.

5.1 Kalibrace kamery v University of Arizona

Při psaní vlastní implementace v Pythonu bylo nejprve nutné ověřit, že dává podobné výsledky kalibrace kamery v University of Arizona jako Lalondeho implementace v Matlabu.

Rozdíl mezi výsledky kalibrace se stejnými parametry v Matlabu a v Pythonu je menší než 3°, což se dá považovat za úspěšné přepsání metody z Matlabu do Pythonu, protože určitý rozdíl výsledků je způsoben použitím odlišných knihoven pro optimalizaci. Přesné hodnoty jsou uvedeny v tabulce 5.1 na prvním a druhém řádku.

5.1.1 Využití Perezova modelu závisejícího na azimutu pro kalibraci zenitu a ohniskové vzdálenosti

Dále bylo provedeno srovnání, jak funguje kalibrace ohniskové vzdálenosti a zenitu pomocí běžného Perezova modelu závislého na pozici slunce. Jelikož ale pozice slunce při kalibraci pomocí datasetu $\mathcal{I}$ není známa, jsou nastaveny výchozí hodnoty tak, aby byl vliv slunce na model co nejmenší. Azimut kamery je nastaven na 0°, azimut slunce na 120°, zenit slunce na 80°. Parametry byly zvoleny podle Lalonde a kol. (2010), Appendix C.

Tato konfigurace odhadla zenit o 3° menší a azimut o méně než 2° menší než výchozí konfigurace s využitím zjednodušeného modelu, což jsou zanedbatelné rozdíly. Větší vliv tato změna ale měla na odhad ohniskové vzdálenosti, která s využitím obecnějšího modelu oblohy je výrazně vyšší. Zorný úhel je tak predikován nižší, místo původních 45° pouze 36°. Pravděpodobně je to kvůli vlivu slunce na modely oblohy na okraji snímku. Reálné snímky oblohy jsou více jednolité než modelované snímky s nízkou ohniskovou vzdáleností, na kterých je na okraji patrný vliv slunce. Optimalizační algoritmus to vyřeší tím, že zvýší ohniskovou vzdálenost, snímky přiblíží a udělá je tak více jednolité s menším gradientem.

5.1.2 Využití Pražského modelu oblohy

Když se pro kalibraci zenitu a ohniskové vzdálenosti použije zjednodušený Perezův model, liší se hodnoty azimutu predikované Pražským modelem a Perezovým modelem o méně než 3°. Pro kalibraci ohniskové vzdálenosti se však PSM neosvědčil, protože predikuje skoro dvakrát větší hodnotu než Perez a zorný úhel

Jazyk	Model I	Model J	$f_c[px]$	ϕ_c	FoV	θ_c
Matlab	Perez Simple*	Perez	894.45	1.51°	43.85°	84,91°
Python	Perez Simple	Perez	867.21	1.20°	45.06°	85,67°
Python	Perez	Perez	1107.18	2.60°	36,02°	82,69°
Python	Perez Simple	PSM**	867.21	3.78°	45.08°	85,67°
Python	PSM	PSM	1547.98	6.50°	26.18°	83,86°

**Perezův model oblohy nezávislý na azimutu, **Prague Sky Model*

Tabulka 5.1: Srovnání kalibrací kamery v University of Arizona, kde f_c je ohnisková vzdálenost, FoV zorný úhel, ϕ_c azimut a θ_c zenit.

místo 45° udává pouze 26°. Rozdíl v zenitu jsou zanedbatelné 2°. Přesné hodnoty jsou v tabulce 5.1 na posledních dvou řádcích.

Bez znalosti skutečného zorného pole kamery je těžké zhodnotit správnost kalibrace, ale pravděpodobně budou správnější hodnoty blíže 60° než hodnoty okolo 25°, které dává Pražský model oblohy.

Z tohoto důvodu bude v dalších experimentech použit Pražský model oblohy pouze na kalibraci azimutu a kalibrace zbylých parametrů bude provedena zjednodušeným Perezovým modelem.

5.1.3 Spuštění experimentů

V Matlabu experiment proběhl spuštěním *demoSkyCalib.m*. Experimenty v Pythonu byly provedeny následujícími příkazy:

```
> docker-compose run bash
> python3 main.py \
   -I ../webcamCalibration/images/gradient  \
   -J ../webcamCalibration/images/clearDay  \
   -m ../webcamCalibration/skyMask/mask.jpg \
   -n 1000 -W 720 -H 540 -f0 500 -df matlab-flat \
   -mI perez-azimuth-independent -mJ perez \
   -min 10 -max 240
...
> python3 main.py \
   -I ../webcamCalibration/images/gradient  \
   -J ../webcamCalibration/images/clearDay  \
   -m ../webcamCalibration/skyMask/mask.jpg \
   -n 1000 -W 720 -H 540 -f0 500 -df matlab-flat \
   -mI perez-azimuth-independent -mJ prague \
   -min 10 -max 240
...
> python3 main.py \
   -I ../webcamCalibration/images/gradient  \
   -J ../webcamCalibration/images/clearDay  \
   -m ../webcamCalibration/skyMask/mask.jpg \
   -n 1000 -W 720 -H 540 -f0 500 -df matlab-flat \
   -mI perez -mJ perez -min 10 -max 240
...
> python3 main.py \
   -I ../webcamCalibration/images/gradient  \
   -J ../webcamCalibration/images/clearDay  \
   -m ../webcamCalibration/skyMask/mask.jpg \
   -n 1000 -W 720 -H 540 -f0 500 -df matlab-flat \
```

```
    -mI prague -mJ prague -min 10 -max 240
...
```

5.2 Kalibrace kamer ČHMÚ z pozice slunce

Na 10 vybraných kamerách ČHMÚ byly nejdříve ručně označeny pozice slunce na obrázcích, u nichž byly známy místo, datum a čas pořízení. Tyto anotace jsou zobrazeny v obrázku 3.5.

Experiment byl spuštěn následujícími příkazy a výsledky byly vráceny na standardní výstup.

```
> docker-compose run bash
...
> root@5b428d7308c9:/usr/bakalarka/src# python3 sun_geometry.py
# doba běhu 41 minut (AMD Ryzen 5 4500U, 8GB RAM)
```

Správnost výsledků lze ověřit pouze orientačně a jestli nabývají smysluplných hodnot. Výsledný azimut lze srovnat s orientací udávanou ČHMÚ. U zenitu a zorného úhlu lze pouze zkontrolovat, jestli jsou v realistickém rozmezí. Výsledné hodnoty azimutu se vesměs shodují s orientací ČHMÚ, kromě kamery v Nedvězí, která je orientována na jih, ale kalibrace říká, že má azimut 88°, tedy že je orientována na východ. Také hodnota v Polomu příliš neodpovídá, ČHMÚ udává západní orientaci, ale kalibrace říká azimut 144°, což je spíše jihovýchod. Může to být způsobené tím, že v tomto případě jsou snímky slunce pořízeny krátkou dobu od sebe, což způsobuje, že označené pozice slunce jsou velmi blízko sebe a program je za těchto okolností citlivější na šum a na nepřesné označení středu slunce. Všechny výsledky kalibrace s využitím principu Perspective-n-Point jsou zaneseny v tabulce 5.2.

Kamera	**Orientace**	ϕ_c	θ_c	FoV
Brno	45°	76.30°	69.17°	52.03°
Č. Budějovice	270°	269.88°	85.92°	53.94°
Dukovany	135°	131.76°	85.17°	58.20°
Holešov	270°	267.50°	85.38°	52.22°
Nedvězí	180°	88.33°	77.04°	45.68°
Olomouc	180°	163.31°	79.08°	52.32°
Polom	270°	144.00°	84.08°	52.32°
Přibyslav	225°	260.70°	84.77°	51.68°
Přimda	90°	79.63°	77.86°	31.70°
Temelín	135°	140.40°	80.26°	49.63°

Tabulka 5.2: Výsledky kalibrace kamer ČHMÚ z pozice slunce. Orientace od ČHMÚ je udávaná s přesností ±22.5°.

5.3 Kalibrace kamer ČHMÚ s modely oblohy

V této sekci jsou představeny výsledky kalibrace vybraných kamer ČHMÚ s využitím Pražského modelu oblohy a Perezova modelu oblohy. Nejprve byly

provedeny na větších méně čistých datasetech $\mathcal{I}_l$ a $\mathcal{J}_l$ a následně i na menších množinách obrázků $\mathcal{I}_s$ a $\mathcal{J}_s$, které byly lépe očištěny o nežádoucí atmosférické jevy. Kalibrace je provedena na dvou velikostech datasetů z toho důvodu, aby byla zjištěna robustnost metody a pro zjištění, s jakým množstvím dat kalibrace funguje lépe.

Počty obrázků použitých pro kalibraci jednotlivých kamer jsou zapsány v tabulce 5.3. Náhodný výběr ze snímků využívaných ke kalibraci je na obrázcích 3.2 a 3.3.

Místo kamery	**Orientace**	$\lvert\mathcal{I}_l\rvert$	$\lvert\mathcal{J}_l\rvert$	$\lvert\mathcal{I}_s\rvert$	$\lvert\mathcal{J}_s\rvert$
Brno	45°	53	210	20	80
České Budějovice	270°	33	41	20	19
Dukovany	135°	68	35	20	20
Holešov	270°	49	59	20	19
Nedvězí	180°	65	151	20	41
Olomouc	180°	60	130	20	46
Polom	270°	64	153	20	57
Přibyslav	225°	52	72	20	20
Přimda	90°	45	99	20	20
Temelín	135°	54	104	20	41

Tabulka 5.3: Kamery ČHMÚ vybrané ke kalibraci a velikosti kalibračních datasetů. Orientace udávaná ČHMÚ je s přesností ±22.5°

5.3.1 Lalondeho implementace

Před hodnocením vlastní implementace na datech ČHMÚ napřed vyzkoušíme, jak s méně dokonale čistými daty funguje referenční implementace.

Experiment byl spuštěn v prostředí Matlab ve složce

```
C:\...\prague-sky-model-webcam-calibration\webcamCalibration
```

příkazy

```
>> chmuCalibration('Ismall', 'Jsmall') % menší dataset
...
>> chmuCalibration('I', 'J') % větší dataset
...
```

Statistické charakteristiky výsledků Rozdíly mezi úhly jsou počítány tak, aby absolutní hodnota rozdílu mezi dvěma úhly byla vždy menší nebo rovna 180° a aby bylo zachováno znaménko. Střední hodnoty (udávající bias kalibrační metody) a směrodatné odchylky rozdílů kalibrací na větším a menším datasetu jsou:

- $E_c[\phi_c^{(small)} - \phi_c^{(large)}] = -6.9°$, $\sigma = 53.5°$
- $E_c[\theta_c^{(small)} - \theta_c^{(large)}] = -0.4°$, $\sigma = 1.1°$

		Větší dataset			Menší dataset		
Kamera	ϕ	ϕ_c	θ_c	FoV	ϕ_c	θ_c	FoV
Brno	45°	74.44°	76.23°	25.91°	75.77°	76.23°	36.37°
Č. Budějovice	270°	257.47°	71.58°	28.10°	256.87°	71.58°	28.23°
Dukovany	135°	120.12°	85.37°	22.29°	141.38°	85.37°	36.16°
Holešov	270°	261.73°	69.24°	28.32°	250.15°	69.24°	32.19°
Nedvězí	180°	213.26°	56.01°	49.13°	274.01°	52.21°	52.90°
Olomouc	180°	179.15°	71.48°	19.03°	177.56°	71.48°	26.99°
Polom	270°	325.22°	83.78°	25.16°	323.44°	83.78°	27.42°
Přibyslav	225°	251.08°	69.34°	30.86°	250.79°	69.34°	35.43°
Přimda	90°	278.17°	87.08°	43.61°	121.87°	87.06°	46.13°
Temelín	135°	141.13°	74.52°	8.02°	161.13°	74.52°	19.25°

ϕ je azimut, který ČHMÚ udává kvantizovaný na 8 hodnot s přesností na 45°.

Tabulka 5.4: Výsledky kalibrace kamer ČHMÚ pomocí Lalondeho implementace v Matlabu

- $E_c[FoV_c^{(small)} - FoV_c^{(large)}] = 6°$, $\sigma = 4.3°$

Rozdíly kalibrovaného azimutu od azimutu udávaného ČHMÚ pak mají následující střední hodnoty a směrodatné odchylky:

- $E_c[\phi_c^{(small)} - \phi_c^{(\check{C}HM\acute{U})}] = 5.8°$, $\sigma = 59.5°$
- $E_c[\phi_c^{(large)} - \phi_c^{(\check{C}HM\acute{U})}] = -23.3°$, $\sigma = 31.9°$

Kdyby kalibrace azimutu byly náhodné s uniformním rozdělením, byla by střední hodnota rozdílu 180° a směrodatná odchylka by byla 104°.

Z výsledků v tabulce 5.4 vidíme, že kalibrace azimutu je nejméně přesná. Odhady zenitu a zorného úhlu z kalibrací nad oběma datasety jsou skoro stejné. Stále ale nevíme a nemáme jak zjistit, jaký má odhad FoV a θ bias.

5.3.2 Perezův model v Pythonu

Zde stejně jako v 5.1 srovnáme, jestli kalibrace pomocí programu v Pythonu Perezovým modelem dává podobné výsledky jako Lalondeho implementace v Matlabu.

Experiment byl spuštěn příkazy

```
> docker-compose run -d chmu-perez
  # výstup ukládá do chmu-perez-log.jsonl
  # doba běhu 1 hodina a 17 minut (AMD Ryzen 5 4500U, 8GB RAM)
> docker-compose run -d chmu-perez-clean
  # výstup ukládá do chmu-perez-clean-log.jsonl
  # doba běhu 12 minut (AMD Ryzen 5 4500U, 8GB RAM)
```

Statistické charakteristiky výsledků Obdobně jako v 5.3.1 srovnáme výsledky na větším a menším datasetu a také srovnáme, jak se odlišují od předchozích výsledků v Matlabu (5.3.1). Kompletní výpis výsledků najdeme v tabulce 5.5. Střední hodnoty a směrodatné odchylky rozdílů kalibrací na větším a menším datasetu jsou:

- $E_c[\phi_c^{(small)} - \phi_c^{(large)}] = -18.9°$, $\sigma = 72.8°$
- $E_c[\theta_c^{(small)} - \theta_c^{(large)}] = -0.7°$, $\sigma = 1.4°$
- $E_c[FoV_c^{(small)} - FoV_c^{(large)}] = 5°$, $\sigma = 6.2°$

Rozdíly kalibrovaného azimutu od azimutu udávaného ČHMÚ pak mají následující střední hodnoty a směrodatné odchylky:

- $E_c[\phi_c^{(small)} - \phi_c^{(\check{C}HM\acute{U})}] = 10.78°$, $\sigma = 79.4°$
- $E_c[\phi_c^{(large)} - \phi_c^{(\check{C}HM\acute{U})}] = -8.2°$, $\sigma = 16.8°$

		Větší dataset			**Menší dataset**		
Kamera		ϕ	ϕ_c	θ_c	ϕ	ϕ_c	θ_c
Brno	45°	73.68°	77.96°	26.74°	75.75°	79.66°	37.70°
Č. Budějovice	270°	251.94°	80.39°	29.45°	251.76°	80.15°	29.28°
Dukovany	135°	141.62°	87.20°	26.84°	154.57°	87.22°	40.83°
Holešov	270°	260.43°	79.74°	29.20°	244.48°	78.50°	32.57°
Nedvězí	180°	214.34°	58.52°	53.67°	289.20°	54.85°	58.40°
Olomouc	180°	178.49°	82.74°	22.80°	179.97°	81.54°	31.60°
Polom	270°	278.95°	86.67°	44.61°	329.67°	86.93°	35.34°
Přibyslav	225°	248.72°	79.55°	33.41°	243.14°	78.82°	39.47°
Přimda	90°	81.08°	87.43°	43.73°	278.65°	87.43°	46.36°
Temelín	135°	152.57°	86.21°	19.86°	5.53°	84.49°	28.87°

Tabulka 5.5: Výsledky kalibrace kamer ČHMÚ v Pythonu Perezovým modelem

Výsledky kalibrace Perezovým modelem v Pythonu v tabulce 5.5 a v Matlabu v tabulce 5.4 se už liší více než na kameře v Arizoně (5.1). Rozdíly mezi sloupci v tabulkách 5.4 a 5.5 mají střední hodnoty směrodatné odchylky uvedené v tabulce 5.6. Zenit a zorný úhel se v průměru liší o jednotky stupňů a v jednotkách je i směrodatná odchylka jejich rozdílů. Oproti kalibraci v Matlabu je zde větší rozdíl mezi větším a menším datasetem. Na větších datech zde kalibrace azimutu funguje výrazně lépe. V Matlabu tento rozdíl není takový. Hodnoty azimutu nejvíce podobné hodnotám ČHMÚ jsou vygenerovány v Pythonu na větších datech.

	Větší dataset			**Menší dataset**		
Vlastnost	ϕ_c	θ_c	FoV	ϕ_c	θ_c	FoV
Střední hodnota	14.00°	6.18°	4.99°	1.98°	5.88°	3.94°
Směrodatná odchylka	52.30°	4.42°	5.79°	70.23°	3.69°	3.05°

Vlastnosti rozdílu $\Delta = (výsledky_{python} - výsledky_{matlab})$, $|\Delta| \leq 180°$

Tabulka 5.6: Statistické vlastnosti rozdílů mezi kalibracemi Perezovým modelem v Pythonu a v Matlabu

5.3.3 Pražský model oblohy v Pythonu

Po předchozích ukázkách fungování kalibrace Perezovým modelem v sekcích 5.3.1 a 5.3.2 se podíváme, jestli využití Pražského modelu oblohy pomůže vylepšit kalibrazi azimutu.

Experiment byl spuštěn příkazy

```
> docker-compose run -d chmu-prague
  # výstup ukládá do chmu-prague-log.jsonl
  # doba běhu 4 hodiny a 52 minut (AMD Ryzen 5 4500U, 8GB RAM)
> docker-compose run -d chmu-prague-clean
  # výstup ukládá do chmu-prague-clean-log.jsonl
  # doba běhu 24 minut (AMD Ryzen 5 4500U, 8GB RAM)
```

Střední hodnoty a směrodatné odchylky rozdílů kalibrací v tabulce 5.7 na větším a menším datasetu jsou:

- $E_c[\phi_c^{(small)} - \phi_c^{(large)}] = -22.8°$, $\sigma = 57.8°$
- $E_c[\theta_c^{(small)} - \theta_c^{(large)}] = -0.7°$, $\sigma = 1.4°$
- $E_c[FoV_c^{(small)} - FoV_c^{(large)}] = 5°$, $\sigma = 6.2°$

Rozdíly kalibrovaného azimutu od azimutu udávaného ČHMÚ pak mají následující střední hodnoty a směrodatné odchylky:

- $E_c[\phi_c^{(small)} - \phi_c^{(\check{C}HM\acute{U})}] = -90°$, $\sigma = 74.4°$
- $E_c[\phi_c^{(large)} - \phi_c^{(\check{C}HM\acute{U})}] = -28.8°$, $\sigma = 129.6°$

Z výsledků vidíme, že kalibrace azimutu funguje hůře než s využitím Perezova modelu. Možná příčina může být to, že model využívá výchozí hodnoty parametrů, jako jsou albedo (jak moc světla se odrazí od zemského povrchu) a viditelnost. Tyto výchozí hodnoty ale nemusejí odpovídat skutečným hodnotám v době pořízení všech snímků použitých pro kalibraci. Optimalizace těchto parametrů by ale vyžadovala netriviální úpravu Pražského modelu oblohy, aby při změně těchto parametrů nebylo nutné model reinicializovat a znovu načítat soubor obsahující model velký stovky MB.

Dále vidíme, že kalibrace zorného úhlu FoV a zenitu θ jsou stejné jako při kalibraci Perezovým modelem v Pythonu v 5.3.2, protože v obou případech byl pro jejich kalibraci využit zjednodušený Perezův model. Vidíme také, že kalibrace dvou kamer ani nedoběhly v časovém omezení jedné hodiny na kameru.

Rozdíly mezi výsledky kalibrace Perezovým modelem v tabulce 5.5 a Pražským modelem v tabulce 5.7 mají statistické vlastnosti uvedené v tabulce 5.8.

5.3.4 Shrnutí kalibrací ČHMÚ

Výsledky ukazují, že kalibrace Perezovým modelem v Pythonu byla implementována úspěšně a srovnatelně s původní implementací v Matlabu. Lépe funguje s větším počtem obrázků. Kalibrace azimutu pomocí Pražského modelu je pomalejší a dává horší výsledky. Z experimentů také plyne, že ohniskovou vzdálenost a zenitový úhel je lepší kalibrovat jednodušším modelem oblohy, který není závislý na azimutu.

Kamera	ϕ	Větší dataset ϕ_c	θ_c	FoV	Menší dataset ϕ_c	θ_c	FoV
Brno	45°	—	77.96°	26.74°	167.41°	79.66°	37.70°
Č. Budějovice	270°	190.14°	80.39°	29.45°	189.34°	80.15°	29.28°
Dukovany	135°	223.67°	87.20°	26.84°	224.60°	87.22°	40.83°
Holešov	270°	14.69°	79.74°	29.20°	186.43°	78.50°	32.57°
Nedvězí	180°	—	58.52°	53.67°	326.24°	54.85°	58.40°
Olomouc	180°	73.53°	82.74°	22.80°	81.38°	81.54°	31.60°
Polom	270°	255.63°	86.67°	44.61°	243.42°	86.93°	35.34°
Přibyslav	225°	187.89°	79.55°	33.41°	183.43°	78.82°	39.47°
Přimda	90°	185.99°	87.43°	43.73°	184.72°	87.43°	46.36°
Temelín	135°	10.90°	86.21°	19.86°	58.87°	84.49°	28.87°

**Prázdná pole znamenají, že kalibrace nedoběhla, neboť byla po 1 hodině běhu přerušena.*

Tabulka 5.7: Výsledky kalibrace kamer ČHMÚ v Pythonu Pražským modelem oblohy

Vlastnost	Větší dataset ϕ_c	θ_c	FoV	Menší dataset ϕ_c	θ_c	FoV
Střední hodnota	-63.59°	0.00°	0.00°	-8.17°	0.00°	0.00°
Směrodatná odchylka	112.36°	0.00°	0.00°	113.31°	0.00°	0.00°

Vlastnosti rozdílu $\Delta = (výsledky_{Prague} - výsledky_{Perez})$, $|\Delta| \leq 180°$

Tabulka 5.8: Statistické vlastnosti rozdílů mezi kalibracemi Perezovým a Pražským modelem

Zenit a ohniskovou vzdálenost by bylo možné kalibrovat i Pražským modelem oblohy, kdyby se provedlo jeho zjednodušení, aby nezávisel na azimutu kamery. Model by v takovém případě mohl vracet průměrnou hodnotu přes všechny azimuty kamery. Vyšší přesnosti by bylo dosaženo i díky tomu, že by měl oproti Lalondeho zjednodušené verzi Perezova modelu navíc k dispozici informaci o pozici slunce. Tyto výsledky by bylo nutné předpočítat, neboť průměrování všech hodnot je náročná operace. Kalibrace by se dala vylepšit také tím, že by se modeloval nejen jas (černobílý obrázek), ale i barvy. Aby parametry modelu (albedo, viditelnost) lépe odpovídaly skutečnosti v místě a čase pořízení každého snímku, daly by se tyto parametry kalibrovat společně s (f, θ, ϕ). To by ale vyžadovalo netriviální úpravu Pražského modelu, aby se daly albedo a viditelnost upravovat za běhu, tedy aby se při každé změně těchto parametrů nemusel model reinicializovat.

5.3.5 Pokus o spuštění DeepCalib

Výsledky kalibrací ohniskové vzdálenosti a zenitu by také bylo vhodné srovnat s kalibrací neuronovými sítěmi DeepCalib. Implementaci DeepCalib se však nepodařilo spustit kvůli nefunkční demo verzi na webu autora.

`https://lvsn.github.io/deepcalib/`

Lokálně se program nepodařilo spustit, protože program byl ukončen neošetřenou výjimkou, která nastala i když program správně načetl natrénované váhy

modelu a byly nainstalovány všechny potřebné knihovny. Program nebyl úspěšně zprovozněn také kvůli málo detailní vývojové dokumentaci.

Závěr

V rámci této práce byly představeny metody kalibrace kamery pomocí vzhledu oblohy, pomocí pozice slunce (Lalonde a kol. 2010) a pomocí neuronových sítí (Hold-Geoffroy a kol. 2022). Dále byla úspěšně napsána implementace kalibrace meteorologických webových kamer pomocí pozice slunce. Implementována byla i metoda využívající Perezův model oblohy (Perez a kol. 1993), do které byl úspěšně zakomponován i Pražský model oblohy (Wilkie a kol. 2021).

Správnost vlastní implementace v Pythonu byla ověřena na datech z webkamery v University of Arizona, která jsou součástí originální implementace v Matlabu jako ukázka metody. Zde kalibrace v Pythonu oběma modely vrací podobné výsledky jako původní implementace v Matlabu. Malý rozdíl je způsoben pouze použitím jiné knihovny pro optimalizaci. Původně je ohnisková vzdálenost a zenit kalibrován zjednodušeným Perezovým modelem nezávislým na azimutu. Při pokusu kalibrovat tyto hodnoty původním Perezovým modelem a Pražským modelem se ale ukázalo, že tyto modely pro kalibraci ohniskové vzdálenosti a zenitu nejsou vhodné. Proto byl Pražský model oblohy využit pouze pro kalibraci azimutu.

Dále byly prováděny experimenty na deseti vybraných meteorologických webových kamerách Českého hydrometeorologického ústavu, na kterých bylo možné provést kalibraci jak podle vzhledu oblohy, tak podle pozice slunce. ČHMÚ však neposkytuje skutečné parametry těchto kamer, pouze hrubou orientaci na světovou stranu (sever, severovýchod, východ, ...). Kalibrace proto bylo možné srovnat pouze mezi sebou a s tímto hrubým odhadem azimutu.

Metoda kalibrace podle pozice slunce využívající princip Perspective-n-Point, která počítá parametry kamery ze srovnání čtyř pozic slunce na obrázku a v prostoru, byla na datech ČHMÚ úspěšná. Odhady azimutu byly blízké hodnotám udávaným ČHMÚ a odhady zenitu a ohniskové vzdálenosti byly realistické.

Kalibrace Perezovým modelem oblohy úspěšně kalibruje zenit, jehož odchylky jsou do 6°. Ohniskovou vzdálenost systematicky vrací příliš velkou, což může být způsobeno tím, že model má pevně nastavené parametry, které by pro každý snímek musely být různé a tak model neodpovídá realitě. Azimut je úspěšně kalibrován pouze Perezovým modelem, Pražským modelem nikoliv. To může být způsobeno tím, že složitější Pražský model lépe odpovídá nastaveným atmosférickým parametrům (viditelnost, albedo), které jsou ale pro jednoduchost nastaveny pevně a nemusejí odpovídat realitě v místě kamery a v čase pořízení snímku. Aby bylo možné kalibrovat i viditelnost a albedo, bylo by nutné upravit implementaci Pražského modelu oblohy, aby podporovala změny těchto parametrů za běhu bez nutnosti reinicializace. Modelování barevných snímků by také mohlo přinést zlepšení.

Výhodou kalibrace z pozice slunce je, že podle srovnání funguje nejlépe. Nevýhodou je, že ne všechny kamery jsou schopné zachytit pozici slunce (např. kamery orientované na sever). Označení pozice slunce na obrázku bylo prováděno ručně, protože bylo kalibrováno pouze 10 kamer. Automatická anotace pro využití na více kamerách by mohla být možná a pomohlo by, kdyby měla k dispozici HDR (High Dynamic Range) snímky, na kterých by slunce mělo výrazně vyšší hodnoty než zbytek oblohy.

Kalibrace modelem oblohy sama o sobě není ideální pro využití v České republice, protože vyžaduje úplně čistou oblohu, což v ČR není úplně obvyklé. Metoda je celkově velmi citlivá na vstupní data. Navíc snímky čisté oblohy obsahující horizontální gradient způsobený sluncem pro kalibraci azimutu je podle Lalondeho nutné z větší části vybírat ručně. Metoda by se dala vylepšit předzpracováním snímků, které by zahrnovalo např. segmentaci oblačnosti a dalších rušivých elementů, aby byla data vhodnější pro kalibraci. Lepších výsledků by se také dalo dosáhnout přesnější parametrizací modelů oblohy. Tyto parametry by se mohly také stát předmětem kalibrace, nebo by se mohly určovat v rámci předzpracování dat pro kalibraci.

Kalibrace pomocí hlubokých neuronových sítí zmíněná v práci nedokáže kalibrovat azimut, ale funguje i na nestatických kamerách a má tedy jiné využití.

Nejvhodnější metodou pro kalibraci meteorologických webových kamer, kterým jejich orientace dovoluje pořídit snímky slunce, se jeví kalibrace podle pozice slunce, které by se dalo označovat automaticky. Kalibrace ostatních meteorologických webových kamer by se poté dala provést podle vzhledu oblohy za využití předzpracovaných snímků s vysegmentovanou mlhou, párou a oblačností.

Seznam použité literatury

360 Cities Holding B.V. (2023). 360Cities. https://www.360cities.net/. Accessed: 2023-04-19.

Chaos Czech a.s. (2023). Corona Renderer. https://corona-renderer.com/. Accessed: 2023-04-17.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. a Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Famfulík, L. (2008). Digitální kyvadlový inklinometr. bakalářská práce, Vysoké učení technické v Brně. URL https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=9490.

Hold-Geoffroy, Y., Piché-Meunier, D., Sunkavalli, K., Bazin, J.-C., Rameau, F. a Lalonde, J.-F. (2022). A deep perceptual measure for lens and camera calibration.

Hoško, J. (2014). Bezpečnostní kamerové systémy. diplomová práce, České vysoké učení technické v Praze. URL https://dspace.cvut.cz/handle/10467/24416.

Huang, G., Liu, Z., van der Maaten, L. a Weinberger, K. Q. (2017). Densely connected convolutional networks. pages 2261–2269. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2017.html#HuangLMW17.

Kingma, D. P. a Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kullback, S. a Leibler, R. A. (1951). On information and sufficiency. *Ann. Math. Statist.*, **22**(1), 79–86.

Lalonde, J.-F., Narasimhan, S. G. a Efros, A. A. (2010). What do the sun and the sky tell us about the camera? *International Journal on Computer Vision*, **88**(1), 24–51.

Levenberg, K. (1944). A method for the solution of certain non – linear problems in least squares. *Quarterly of Applied Mathematics*, **2**, 164–168.

Mirbauer, M., Rittig, T., Iser, T., Kriváneк, J. a Šikudová, E. (2022). SkyGAN: Towards Realistic Cloud Imagery for Image Based Lighting. In *Eurographics Symposium on Rendering*. The Eurographics Association. ISBN 978-3-03868-187-8. doi: 10.2312/sr.20221151. URL https://diglib.eg.org:443/xmlui/handle/10.2312/sr20221151.

Nikon (2023). Understanding Focal Length. https://www.nikonusa.com/en/learn-and-explore/a/tips-and-techniques/understanding-focal-length.html. Accessed: 2023-04-17.

Perez, R., Seals, R. a Michalsky, J. (1993). All-weather model for sky luminance distribution—preliminary configuration and validation. *Solar Energy*, **50**(3), 235–245. ISSN 0038-092X. doi: https://doi.org/10.1016/0038-092X(93)90017-I. URL https://www.sciencedirect.com/science/article/pii/0038092X9390017I.

Wikipedia (2023). Spectral sensitivity. URL https://en.wikipedia.org/wiki/Spectral_sensitivity#/media/File:Cones_SMJ2_E.svg. Accessed: 2023-04-19.

Wikipedia (2023). Aircraft Principal Axes. https://en.wikipedia.org/wiki/Aircraft_principal_axes#/media/File:Yaw_Axis_Corrected.svg. Accessed: 2023-04-18.

Wikipedia (2023). Sférická soustava souřadnic. URL https://cs.wikipedia.org/wiki/Sf%C3%A9rick%C3%A1_soustava_sou%C5%99adnic#/media/Soubor:Spherical_with_grid.svg. Accessed: 2023-04-17.

Wilkie, A., Vevoda, P., Bashford-Rogers, T., Hošek, L., Iser, T., Kolářová, M., Rittig, T. a Křivánek, J. (2021). A fitted radiance and attenuation model for realistic atmospheres. *ACM Trans. Graph.*, **40**(4). ISSN 0730-0301. doi: 10.1145/3450626.3459758. URL https://doi.org/10.1145/3450626.3459758.

Český hydrometeorologický ústav (2023). Mapa webových kamer. https://www.chmi.cz/files/portal/docs/meteo/kam/. Accessed: 2023-04-18.

Seznam obrázků

Seznam tabulek

A. Přílohy

A.1 Zdrojový kód a data

Repozitář ve složce `src` obsahuje zdrojový kód v Pythonu. Ve složkách `data` a `data-matlab` jsou demonstrační data a ve složce `results` výsledky. Základní informace o programu jsou uvedeny v souboru `README.md`. Repozitář je k dispozici také na následující adrese.

`https://gitlab.mff.cuni.cz/krsickao/prague-sky-model-webcam-calibration`

Upravený Lalondeho kód v Matlabu se nachází ve složce `webcamCalibration` pouze na výše zmíněné adrese, nikoliv v elektronické příloze této práce.