

STUDENTSKÝ FAKULTNÍ PROJEKT

# Barevný posun při relativistických rychlostech

Autor:

*Jakub Novotný*

Vedoucí projektu:  
MGR. PAVEL STRÁNSKÝ, PH.D.

Vypracování:  
*1. leden až 30. duben 2019*

6. května 2019

# Obsah

<b>Úvod</b>	<b>2</b>
<b>1 Úvod do práce s barevami</b>	<b>3</b>
1.1 Lidské vnímaní barev . . . . .	3
1.2 Gamut . . . . .	4
1.3 Barevné prostory . . . . .	6
<b>2 Zpracování barev</b>	<b>7</b>
2.1 Spektrální rozklad . . . . .	7
2.2 Diskrétní spektrum . . . . .	8
2.3 Fialovočervená linie . . . . .	9
2.4 Ze spektra na RGB . . . . .	9
<b>3 Relativistický Dopplerův jev</b>	<b>11</b>
<b>4 Struktura programu</b>	<b>14</b>
4.1 Pomocné funkce . . . . .	14
4.2 Funkce Spektr . . . . .	14
4.3 Program . . . . .	19
4.4 Ukázky . . . . .	28
<b>5 Závěr</b>	<b>35</b>
<b>Literatura</b>	<b>35</b>

# Úvod

Na začátku byla snaha vytvořit jednoduchý program, do kterého by uživatel nahrál obrázek, zvolil by rychlosť v relativistické oblasti a program by již dopočítal novou podobu obrázku po dopplerovské posunutí barev. Barva obrázku je běžně v počítači v reprezentaci nějakého barevného prostoru (nejznámějším je pravděpodobně RGB). Jak však tuto reprezentaci převést na vlnovou délku nebo frekvenci, pomocí kterých se vypočítá posunutí při Dopplerově jevu?

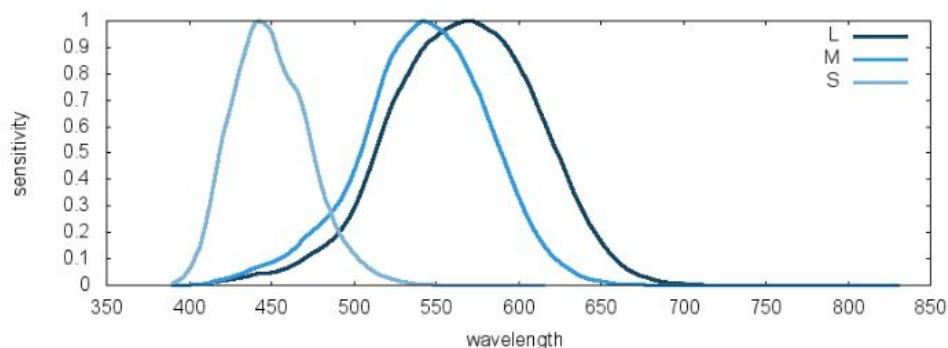
Často používané metody navrhují určit dominantní vlnovou délku, případně převést složky RGB na červenou, zelenou a modrou spektrální barvu. Tyto téměř monochromatické reprezentace nejsou vhodné při práci se snímky našeho světa, který je osvětlen bílým světlem. Pro dobré přiblížení efektru Dopplerova jevu je podstatný charakter spektra, které bude posouváno. Při běžném osvětlení polychromatickým světlem jsou tak spektra pozorovaných barev člověkem spíše spojitá. To je motivací k vytvoření nové metody získání světelného spektra z barvy reprezentované v RGB souřadnicích. Není snahou získat přesné spektrum, které daný objekt původně vyzařoval. Podstatné je docílit jeho spojitého charakteru, díky kterému získáme lepší přiblížení při dopplerovském posunutím.

# Kapitola 1

## Úvod do práce s barvami

### 1.1 Lidské vnímaní barev

Člověk vnímá světlo pomocí světločidných buněk v oku. Citlivé na vlnovou délku dopadajícího světla jsou čípky. Rozlišujeme tři typy čípků S,M a L, kdy každý z nich je citlivý na jinou oblast vlnových délek (obr. 1.1). Nejcitlivější jsou pak postupně na oblasti kolem modré, zelené a červené barvy [1].



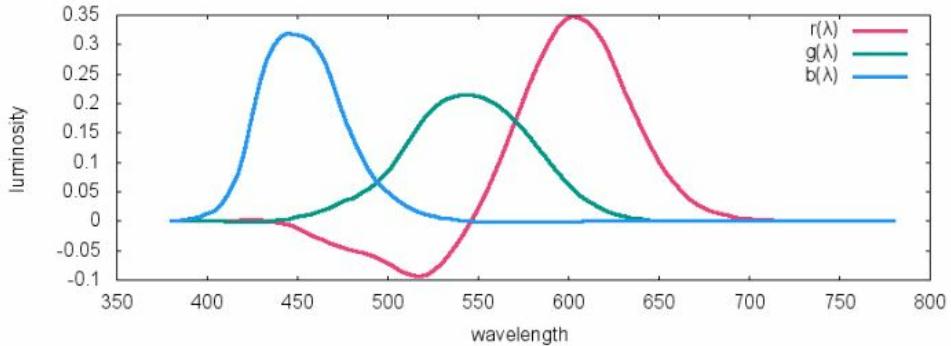
Obrázek 1.1: Citlivost jednotlivých čípků pro vlnovou délku dopadajícího světla. Zdroj [1].

Barevný vjem vzniká na základě stimulace čípků dopadajícím světelným spektrem. Převod mezi spektrem a barevným vjemem není prosté zobrazení a spektra s různým průběhem mohou stimulovat čípky stejným způsobem. To nám však dovoluje používat obrazovky, kterým k zobrazení velkého počtu barev (jak se dále ukáže nelze zobrazit všechny) stačí jen soustava tří barevných diod. Digitální záznam barevného světla proto funguje na podobném principu jako lidské oko a výsledná informace o dopadajícím světle se skládá ze tří hodnot odpovídajících stimulaci tří druhů oblastí čipu, které jsou citlivé nejvíce v oblasti vlnových délek kolem modré, červené a zelené.

Prvním zásadním bodem je tedy zjištění, že různá spektra mohou působit stejný barevný dojem a po zaznamenání spektra ať už člověkem, nebo digitálně se informace o přímém průběhu spektra ztrácí. Najít metodu, která by barvu pixelu převedla na dané spektrum, které bylo zaznamenáno, tak není možné. Cílem však není nalézt přesné spektrum, ale spektrum, které by svým charakterem odpovídalo spektrům reálným.

Možnost zreprodukrovat libovolné barvy pouze pomocí zeleného, modrého a červeného světla byla lákavá. Byly tedy změřeny funkce na obrázku 1.2, které udávají poměr červeného, modrého

a zeleného světla, který zreprodukuje světlo o dané vlnové délce.



Obrázek 1.2: Barevné funkce pro získání barvy světla, která je člověkem vnímána jako totožná se spektrální barvou o příslušné vlnové délce. Zdroj [1]

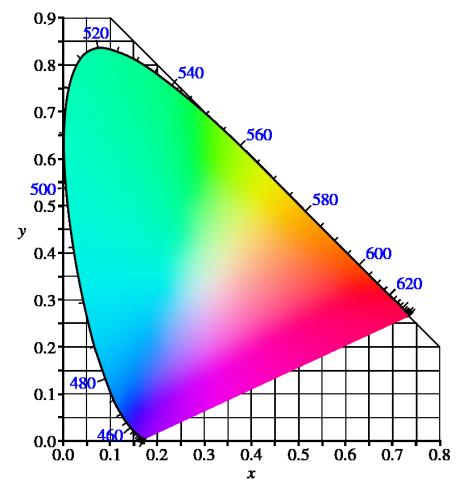
Barvu světla každé vlnové délky je tak možné vyjádřit pomocí tří hodnot R,G a B (jedná se o normované hodnoty funkcí z obrázku 1.2 intenzitou dané složky, neboť různé typy čípků jsou různě citlivé, například zelená barva se při nižší dopadající intenzitě jeví stejně výrazná jako červená; více v [9].). To nám dovoluje reprezentovat barvy pomocí souřadnic ve třírozměrném prostoru. O tom, jaké prostory se volí, resp. jaké báze, více v části *Barevné prostory*. Pro nás je nyní podstatné, že pokud normujeme jednotlivé složky R,G,B intenzitou ( $R+G+B$ ) dané barvy, dostaneme složky  $r,g,b$ , které nejsou závislé na intenzitě a jejich součet je jedna. Libovolnou složku tak může vyjádřit pomocí zbylých dvou. Souřadnice dané barvy tedy budou dány polohou ve dvourozměrném prostoru (rovině) a intenzitou. Takový prostor se nazývá *xyY barevný prostor* a útvar, který reprezentuje barvy v rovině  $xy$  se nazývá *gamut* [2]. Ten je natolik zásadní pro naši práci, že si zaslouží vlastní sekci.

## 1.2 Gamut

Gamut je znázorněn na obrázku 1.3. Jedná se o rovinu  $Y = \text{konst.}$  v barevném prostoru  $xyY$ . Veškerá informace o chromacitě dané barvy je tak dáná souřadnicemi  $x,y$ .

Okraj gamutu tvoří spektrální barvy. Ve vnitřní oblasti se pak nacházejí namíchané barvy. Princip míchání barev je v gamutu velmi názorný. Vezmeme-li dvě barvy, tedy dva body v gamutu, pak barvy, které mohou vzniknout jejich přímým smícháním se nacházejí na jejich spojnici. Gamut je konvexní útvar, spojnice každých dvou bodů se tak nacházejí v gamutu, a tedy každou barvu lze vyjádřit pomocí libovolné jiné barvy a příslušné barvy do páru. Obecný vzorec pro určení souřadnic  $x_{mix}, y_{mix}$  barvy, která vznikne složením  $n$  barev o souřadnicích  $x_i, y_i$  a intenzitě  $L_i$  je tento [3]

$$x_{mix} = \frac{\sum_{i=1}^n \frac{x_i}{y_i} L_i}{\sum_{i=1}^n \frac{L_i}{y_i}}, \quad y_{mix} = \frac{\sum_{i=1}^n L_i}{\sum_{i=1}^n \frac{L_i}{y_i}}. \quad (1.1)$$



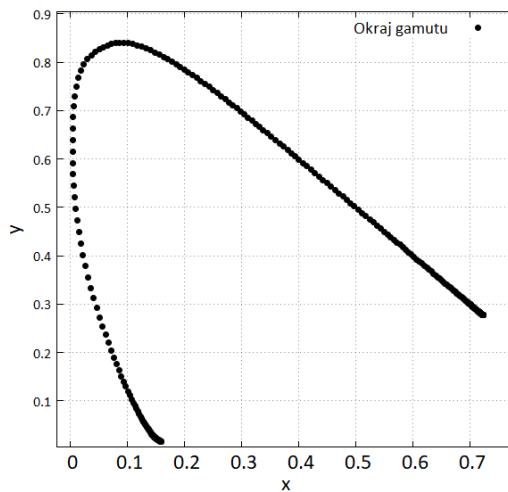
Obrázek 1.3: Gamut v rovině  $xy$  prostoru  $xyY$ . Zdroj [3].

Fakt, že k vytvoření libovolné barvy je zapotřebí pouze pár spektrálních barev s příslušnou vlnovou délkou a to, že každá spektrální barva s vhodnou partnerskou spektrální barvou může vytvořit libovoulou barvu, jsou základem pro naši metodu převodu barvy pixelu na spektrum. Více je probrána v kapitole 2.

Podstatný je také poměr intenzit  $L_i$  v barevném mixu dílčích barev. Pokud bychom skládali pouze dvě barvy o souřadnicích  $[x_1, y_1]$  a  $[x_2, y_2]$ , pak pro jejich intenzity  $L_1$  a  $L_2$ , kterými jsou v daném mixu zastoupeny, platí [3]

$$\frac{L_1}{L_2} = \frac{y_1(x_2 - x_{mix})}{y_2(x_{mix} - x_1)} \quad (1.2)$$

Jak už bylo řečeno, spektrální barvy se nacházejí na okraji gamutu. Můžeme tedy získat datový soubor se souřadnicemi  $xy$  v prostoru  $xyY$  jednotlivých spektrálních barev. Data, se kterými budeme pracovat jsou znázorněna na obrázku 1.4



Obrázek 1.4: Data, která definují okraj gamutu, se který budeme pracovat při odečtení spektra z barvy pixelu. Jejich zdrojem je [4].

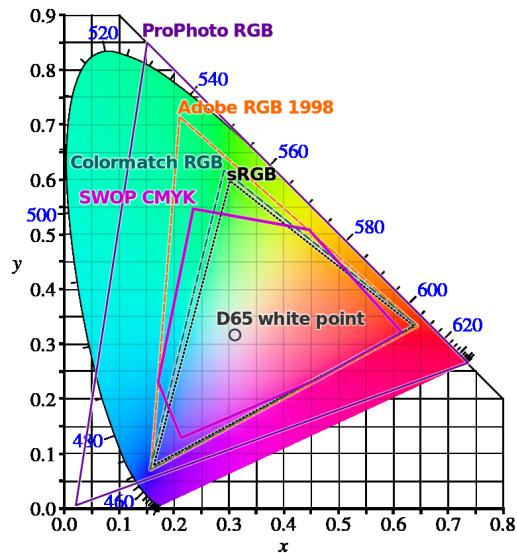
Nyní se zaměříme ještě na zobrazení barev. Bavili jsem se o tom, že danou barvu lze zreproduktovat smícháním zeleného, modrého a červeného světla. Z gamutu je však patrné, že smícháním tří barev jsme schopni zreproduktovat pouze barvy uvnitř jimi daným trojúhelníkem. Vzniká tak otázka, jakou trojici barev vybrat, tedy jaký barevný prostor s barevnou bází zvolit .

### 1.3 Barevné prostory

Bází barevného prostoru  $xyY$  jsou chromatické souřadnice barvy v gamutu a její intenzita.

Existují ještě další barevné prostory s bázemi, které byly voleny podle různých kritérií [5]. Mezi reprezentacemi v různých bázích lze však přecházet pomocí daných matic přechodu. Základním barevným prostorem je prostor  $XYZ$ , ze kterého byl následně odvozen prostor  $xyY$ . Souřadnice  $Y$  opět udává intenzitu dané barvy. Chromatické souřadnice složek  $XYZ$  tvoří vrcholy trojúhelníku, který obsahuje celý gamut. Jejich smícháním tak mohou vzniknout veškeré reálné barvy. Samy se však nacházejí mimo gamut a reálným barvám neodpovídají (obr. 1.5).

Pro reálnou reprodukci barev je však nutné volit vrcholy trojúhelníku jako skutečné barvy a tedy uvnitř gamutu. Dochází tak k další ztrátě informace, neboť původní barevný prostor je zúžen (obr. 1.5). Barevných prostore vzniklo mnoho a definice byla většinou dána čistě volbou dané firmy, která s prostorem pracovala. Často používaným prostorem je  $sRGB$ .



Obrázek 1.5: Příklad barevných prostoreů. Zdroj [2]

# Kapitola 2

## Zpracování barev

Znalosti z předcházející kapitoly nyní využijeme k vytvoření metody převodu informace o barvě pixelu na barevné spektrum. Také se bude hodit umět převádět spektrum zpět do formátu RGB.

### 2.1 Spektrální rozklad

Základní myšlenka naší nové spektrální metody je jednoduchá a vychází přímo z barevné reprezentace v gamutu:

*Libovolnou barvu lze namíchat z libovolné spektrální barvy o vlnové délce  $\lambda_0$  a jí příslušné partnerky.*

To tedy znamená, že každá spektrální barva s vlnovou délkou v intervalu  $[\lambda_{min}, \lambda_{max}]$ , kde  $\lambda_{min}, \lambda_{max}$  jsou krajní hodnoty viditelného spektra, mohou být zastoupeny ve spektru barvy, kterou rozkládáme. Obecně nemáme důvod k vyloučení nějakých spektrálních barev, proto se zdá rozumné každou z nich do spektra každé rozkládané barvy zahrnout. Nicméně jí musíme zahrnout s příslušnou intenzitou  $L$ .

Definujme proto funkci  $L(\lambda)$ , která každé spektrální barvě o vlnové délce  $\lambda \in [\lambda_{min}, \lambda_{max}]$  přiřadí intenzitu, se kterou je ve spektru rozkládané barvy zastoupena. Tuto funkci získáme následujícím způsobem.

Mějme barvu  $\beta$  se souřadnicemi v prostoru sRGB  $\beta_{sRGB} = [R, G, B]$  (V kapitole 4 se dozvím, že přesně v této podobě dostaneme z počítače informace o barvě daného pixelu). Dále ji pomocí matic přechodu (opět viz kapitola 4) převedeme do prostoru  $xyY$ . Její nové souřadnice jsou  $\beta_{xyY} = [x_0, y_0, Y_0]$ . Hodnota  $Y_0$  udává intenzitu barvy  $\beta$ .

Dále známe v prostoru  $xyY$  křivku  $G(x, y)$ , která tvoří okraj gamutu a spektrální barvě s vlnovou délkou  $\lambda$  přiřazuje souřadnice  $[x, y]$  v prostoru  $xyY$ . Mějme tedy spektrální barvu  $\mu$  s vlnovou délkou  $\lambda_1$  o souřadnicích  $[x_1, y_1]$  v prostoru  $xyY$ . Zajímá nás, s jakou partnerskou spektrální barvou  $\lambda_2$  o zatím neznámých souřadnicích  $[x_2, y_2]$  vznikne smícháním tohoto páru barev, barva  $\beta$ .

Jedná se opět o jednoduchou úlohu, neboť hledáme jen průsečík  $[x_2, y_2]$  přímky, která je dány body  $[x_0, y_0]$  a  $[x_1, y_1]$ , a okraje gamutu  $G(x, y)$ . Získáme tak soustavu rovnic pro  $x$  a  $y$ :

$$(y_{mix} - y_1)x + (x_1 - x_{mix})y + (x_{mix}y_1 - y_{mix}x_1) = 0, y = G(x).^1 \quad (2.1)$$

Funkci  $G(x, y)$  máme zadanou explicitně pomocí jednotlivých bodů, soustavu je proto nejjednodušší řešit na počítači. Řešením je pak bod  $[x_2, y_2]$ , který přísluší barvě  $\nu$  s vlnovou délkou  $\lambda_2$ . Pokud bychom naopak hledali partnerku pro barvu  $\nu$ , byla by jí jistě barva  $\mu$ . Jen s pomocí

---

<sup>1</sup>Okraj gamutu lze lokálně vyjádřit jako funkci jedné proměnné.

barev  $\mu$  a  $\nu$  jsme tak schopni namíchat barvu  $\beta$ . Poměr intenzit  $L_i$ , se kterými jsou v dané barvě zastoupeny, je pak dán vztahem 1.2. Stále však neznáme absolutní hodnoty těchto intenzit.

V našem výsledném spektru bude každé vlnové délce  $\lambda$ , která reprezentuje příslušnou spektrální barvu, příslušet nějaká hodnota intenzity  $L(\lambda)$ . Neuvažujeme koherentní světlo, nečekáme proto žádné interferenční projekty. Z toho plyne normovací podmínka pro naší intenzitní funkci  $L(\lambda)$ :

$$Y_0 = \int_{\lambda_{min}}^{\lambda_{max}} L(\lambda) d\lambda \quad (2.2)$$

Ke každé spektrální barvě o vlnové délce  $\lambda$  z intervalu  $\langle \lambda_{min}, \lambda_{max} \rangle$  najdeme její partnerku. Budeme tak mít nespočetnou množinu dvojic barev. Každé z těchto dvojic přiřadíme hodnotu intenzity  $dL$  (tato hodnota je pro všechny dvojice stejná) tak, že platí

$$Y_0 = \int_0^{Y_0} dL. \quad (2.3)$$

Intenzita příslušející  $dL$  dvojici barev je pak dána součtem jejich intenzit  $L_i$

$$dL = L_1 + L_2. \quad (2.4)$$

Použijem-li nyní vztah 1.2 a jeho pravou stranu označíme jako  $K$ , dostaneme vztahy pro určení intenzit dílčích barev:

$$L_1 = dL \frac{K}{1+K}, L_2 = dL \frac{1}{1+K}, K = \frac{y_1(x_2 - x_{mix})}{y_2(x_{mix} - x_1)} \quad (2.5)$$

Jelikož se každá spektrální barva nachází v naší množině uspořádaných dojic barev, ze kterých jsme míchali barvu  $\beta$ , právě dvakrát (jednou jsme pro ni hledali partnerku a podruhé jsme ji našli, když jsme hledali partnerku k její partnerce), je hodnota intenzity  $L(\lambda)$ , kterou je ve spektru zastoupena dána jako

$$L(\lambda) = 2L_1(\lambda) = 2L_2(\lambda), \quad (2.6)$$

kde  $L_1(\lambda)$  je ze vztahu 2.5, kdy  $\lambda$  byla barva, ke které jsme hledali partnerku a  $L_2(\lambda)$  je případ, kdy byla hledanou partnerkou.

Funkce  $L(\lambda)$  je naším hledaným spektrem pro reálnou barvu  $\beta$ , neboť udává intenzitu každé viditelné spektrální barvy s vlnovou délkou  $\lambda$  tak, že jejich smíčcháním vznikne barva  $\beta$  s původní intenzitou.

## 2.2 Diskrétní spektrum

Spojité spektrum je sice pěkné, ale okraj gamutu máme daný pouze jako explicitní výčet bodů, ktrých je nejen spočetně, ale také dokonce konečné množství. To by se sice dalo obejít a křivku  $G(x, y)$  po částech definovat spojitými funkcemi, ale my se zaměříme na práci s diskrétním spektrem s konečným počtem spektrálních barev. Samotný proces vytváření spektra se naštěstí od situace se spojitým spektrem liší jen v několika bodech které nyní rozebereme.

Mějme tedy  $N$  známých spektrálních barev. Každé z nich odpovídá jeden bod na okraji gamutu. Partnerské barvy opět budeme hledat pomocí průsečíku přímky a okraje gamutu. Průsečík přímky a okraje však nemusí být jedním z  $N$  známých bodů gamutu. Nechť tedy přímka protne okraj gamutu mezi dvěma body sousedních barev se známými vlnovými délками  $\lambda_i$  a  $\lambda_{i+1}$ . Průsečík sice odpovídá nějaké spektrální barvě, její přesnou vlnovou délku neznáme. Proto s ní budeme pracovat jako s ostatními barevami v gamutu a pokusíme se ji namíchat pomocí nejbližších spektrálních barev s vlnovými délky  $\lambda_i$  a  $\lambda_{i+1}$  (proces míchání je stejný jako v první kapitole

vztahy (1.1), (1.2) a (2.5). Obecně můžeme volit hustotu pokrytí okraje gamutu známými body různě. Můžeme volit okraj gamutu tak, aby body známých barev byly rozmístěny ekvidistatně a to buď ekvidistatně co se týká vlnové délky  $\lambda$  (takto je voleno pokrytí i v následujících programech), nebo délky v rovině  $xy$  prostoru  $xyY$ .

Díky diskrétnímu spektru nám také integrály z minulé sekce přechází na konečné sumy a co je nejdůležitější, umíme bez výčitek svědomí přidělit intenzitu jednotlivým dvojicím barev, ze kterých budujeme spektrum barvy  $\beta$  o intenzitě  $Y_0$ , neboť  $dL$  přechází na  $\Delta L$  a platí:

$$\Delta L = \frac{Y_0}{N}. \quad (2.7)$$

Naopak vztah (2.6) se stává složitějším. Protože je daná barva brána jako partnerská při protnutí přímky intervalu a ne jen bodu, je nutné do intenzity příslušné spektrální barvy, kterou přispívá do spektra barvy  $\beta$ , zahrnout. Daná barva mohla být určena jako partnerka pro více spektrálních barev a veškeré tyto příspěvky  $L_2^i$  musíme do výsledné intenzity zahrnout. Byla-li tedy například  $\lambda$  brána k-krát jako partnerská barva, je pak její výsledná hodnota intenzity  $L(\lambda)$  ve spektru dána jako:

$$L(\lambda) = L_1(\lambda) + \sum_{i=1}^k L_2^i(\lambda). \quad (2.8)$$

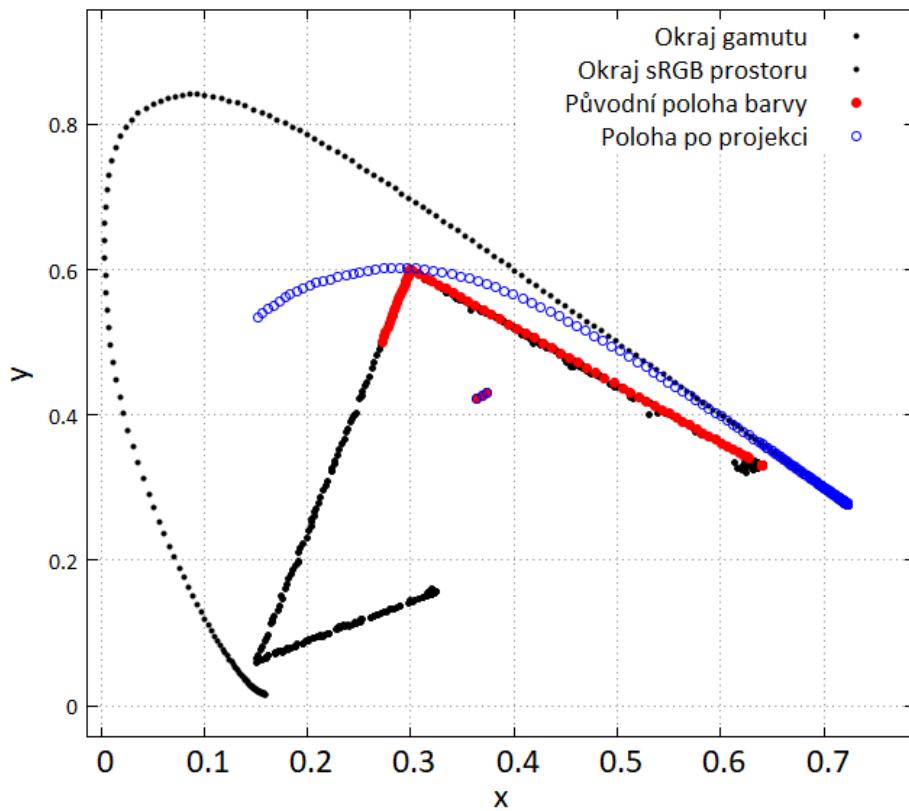
## 2.3 Fialovočervená linie

Fialovočervená oblast okraje gamutu neodpovídá žádné spektrální barvě, ale jedná se o namíchané barvy  $\lambda_{min}$  a  $\lambda_{max}$ . Nalezneme-li v této oblasti partnerskou barvu, budeme s ní pracovat jako s běžnou partnerkou. Na závěr při tvorbě spektra partnerky na fialovočervené přímce musíme rozložit na  $\lambda_{min}$  a  $\lambda_{max}$  a příslušné intenzity jim pak přičíst v sumách (2.8).

## 2.4 Ze spektra na RGB

Mějme obecné spektrum. Pro každou barvu o vlnové délce  $\lambda$  a souřadnicích  $[x, y]$  známe intenzitu  $L$  v reprezentaci prostoru  $xyY$ . Toto spektrum chceme převést na jednu barvu o souřadnicích RGB. Polohu výsledné barvy v prostoru  $xyY$  určíme pomocí vztahu (1.1). Intenzita výsledné barvy je dána jako suma dílčích intenzit. Nyní bychom mohli převést reprezentaci v prostoru  $xyY$  pomocí matic přechodu na vektor v RGB souřadnicích.

Problém však nastane, pokud namíchaná barva nebude ležet v barevném prostoru, se kterým daný programovací jazyk pracuje. Nyní totiž nemáme jištětu, jakým způsobem bude barva reprezentována. Proto je vhodné udělat projekci této barvy na barevný prostor. V reprezentaci  $xyY$  má barevný prostor většinou podobu trojúhelníku v rovině  $xy$ . Leží-li namíchaná barva v tomto trojúhelníku, zobrazí se při projekci sama na sebe. Pokud leží mimo, uděláme kolmou projekci na nejbližší stranu trojúhelníku jak je naznačeno na obrázku 2.1.



Obrázek 2.1: Projekce bodů na prostor sRGB.

Bod dané barvy nejprve ležel v trojúhelníku, který určuje prostor sRGB. Poté byla barva dopperovským posouváním. Přitom došlo k vystoupení z prostoru sRGB. Modré jsou tak naznačeny původní polohy a červeně jejich projekce. Přímky, na kterých leží strany trojúhelníku, určují tři poloroviny ve vnější oblasti. Pokud bod dané barvy leží ve vnější oblasti v průniku dvou polorovin, je zobrazen na vrchol trojúhelníku, který leží zároveň na průniku těchto dvou polorovin. Pokud bod leží pouze v jedné polorovině, je zobrazen na svou kolmou projekci na danou stranu, která vymezuje tuto polorovinu.

## Kapitola 3

# Relativistický Dopplerův jev

K Dopplerovu jevu dochází, pokud se zdroj a přijímač signálu vůči sobě pohybují nenulovou rychlostí. Klasický Dopplerův jev si lze intuitivně objasnit pomocí obrázku 3.1. Zdroj Z vysílá periodický signál s periodou  $T_Z$ , který se šíří prostorem rychlostí  $c$ . Ke zdroji se ve směru šíření signálu přibližuje přijímač P rychlostí  $v$ . Jak je ukázáno na obrázku, přijímač nejprve zachytí minimum. Druhé minimum se však k přijímači přibližuje rychlostí  $v+c$ . Jako periodu  $T_P$  signálu, který změřil přijímač, označíme časový rozdíl mezi naměřením jednotlivých minim. Ta je tak dána jako

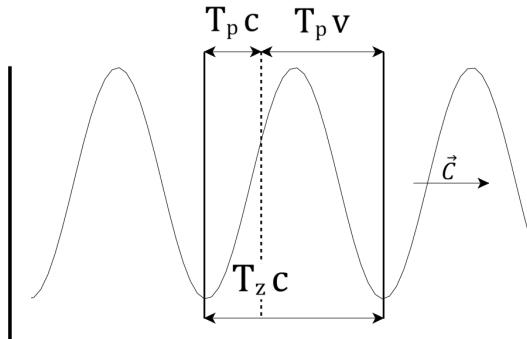
$$T_P = T_Z \frac{c}{c+v}. \quad (3.1)$$

V pohledu speciální relativity [6] je však nutné vztah (3.1) upravit. Mějme nyní obecnou situaci, kdy se přijímač pohybuje v obecném směru vůči světelnému signálu jak je znázorněno na obrázku 3.2. Rychlosť přijímače vůči zdroji je dána jako vektor  $\vec{v}_p$  ve třírozměrném prostoru. Směr šíření signálu je pak dán vlnovým vektorem  $\vec{k}$ , který je také z třírozměrného prostoru. Chceme-li vzít do úvahy relativitu, musíme se na celou situaci dívat ve čtyřrozměrném časoprostoru. Vlnový čtyřvektor  $\mathbf{k}$  se tak skládá z časové složky a prostorových. Časová složka je dána frekvencí signálu  $\omega$  a prostorovou složkou zůstává vektor  $\vec{k}$  (vztah 3.2). Nás bude zajímat, jak se liší čtyřvektor  $\mathbf{k}$  v souřadné soustavě zdroje a v souřadné soustavě přijímače

$$\mathbf{k} = \begin{bmatrix} \omega \\ \frac{c}{\vec{k}} \end{bmatrix}. \quad (3.2)$$

Nejprve si celou situaci rozebereme v soustavách zdroje a přijímače. To je provedeno na obrázku 3.3. První rozbor je v soustavě Z. V této soustavě se pohybuje přijímač a jeho rychlosť je dána jako  $\vec{v}_p$ . Vektory  $\vec{v}_p$  a  $\vec{k}_z$  leží v jedné rovině. Vektor  $\vec{k}_z$  můžeme rozložit na kolmou a rovnoběžnou složku v soustavě Z jako  $\vec{k}_{z\perp}$  a  $\vec{k}_{z\parallel}$ .

V soustavě spjaté s přijímačem P se naopak mění poloha zdroje. Jeho rychlosť je dána vektorem  $\vec{v}_z$ . Opět rozložíme vektor  $\vec{k}_z$  na kolmou a rovnoběžnou složku, nyní však v soustavě P. Dostáváme tak vztahy



Obrázek 3.1: Intuitivní představa klasického Dopplerova jevu. Pozorovatel naměřil periodu signálu  $T_P$ , protože se k signálu blíží rychlosť  $v$ .

$$\begin{aligned}
\vec{k}_z &= k_{z\parallel} \vec{e}_{z\parallel} + \vec{k}_{z\parallel} & \vec{k}_p &= k_{p\parallel} \vec{e}_{p\parallel} + \vec{k}_{p\parallel} \\
k_{z\parallel} &= k_z \cos \theta_z & k_{p\parallel} &= k_p \cos \theta_p \\
\|\vec{k}_{z\perp}\| &= k_z \sin \theta_z & \|\vec{k}_{p\perp}\| &= k_p \sin \theta_p
\end{aligned} \tag{3.3}$$

Nyní můžeme provést Lorentzovu transformaci podle [6] vlnového čtyřvektoru  $\mathbf{k}$  ze soustavy zdroje do soustavy přijímače a podívat se, jak se změní frekvence  $\omega$ . Protože jsme rozkládali vlnový třívektor  $\vec{k}$  do rovnoběžného směru s rychlostí a na ni kolmého podprostoru, zobrazí se jeho kolmé složky  $\vec{k}_\perp$  na sebe pomocí identity  $\mathbf{E}_\perp$ . Transformace má pak tvar

$$\begin{bmatrix} \frac{\omega_p}{c} \\ k_{p\parallel} \\ \vec{k}_{p\perp} \end{bmatrix} = \begin{bmatrix} \cosh \beta & \sinh \beta & 0 \\ -\sinh \beta & \cosh \beta & 0 \\ 0 & 0 & \mathbf{E}_\perp \end{bmatrix} \begin{bmatrix} \frac{\omega_z}{c} \\ k_{z\parallel} \\ \vec{k}_{z\perp} \end{bmatrix} \tag{3.4}$$

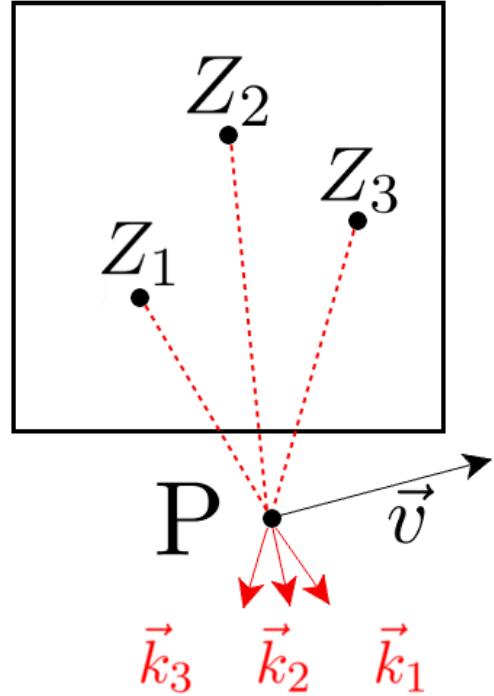
Je vhodné ještě uvést platné vztahy

$$\cosh \beta = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}}, \quad \frac{\sinh \beta}{\cosh \beta} = \frac{v}{c}, \tag{3.5}$$

kde  $v$  je velikost vzájemné rychlosti v tomto případě zdroje a přijímače.

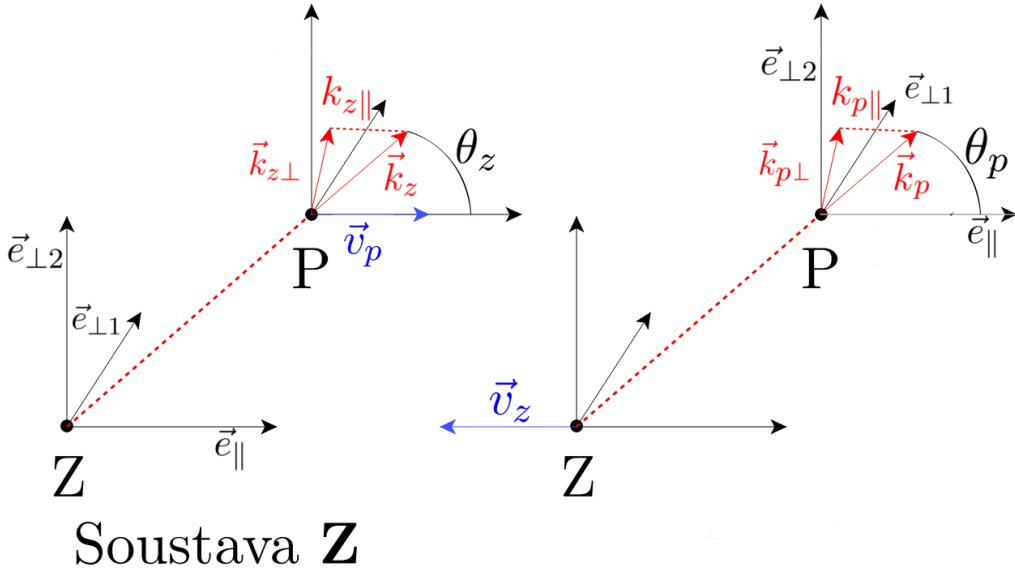
Po provedení transformace a použití vztahů 3.3 a 3.5 dostáváme vztah mezi frekvencemi světelného signálu

$$\frac{\omega_z}{\omega_p} = \frac{\lambda_p}{\lambda_z} = \frac{1 - \frac{v}{c} \cos \theta_z}{\sqrt{1 - \frac{v^2}{c^2}}}. \tag{3.6}$$



Obrázek 3.2: Zdroje vyzařují světelné signály s vlnovými vektory  $\vec{k}$  směrem k přijímači, který se pohybuje s obecným vektorem rychlosti.

## Soustava P



Obrázek 3.3: Rozbor posílání světleného signálu v soustavách zdroje a přijímače ve 3D

Veličiny  $\lambda_p$  a  $\lambda_z$  jsou vlnové délky signálu v soustavě přijímače P a zdroje Z. Poslední neznámou složkou tak zůstává  $\cos \theta_z$ . Úhel  $\theta_z$  svírají vektory  $\vec{v}_p$  a  $\vec{k}_z$  reprezentované v soustavě zdroje. V naší úloze budeme znát směr i velikost rychlosti pozorovatele i jeho polohu vůči obrázku. Zdroji světelného signálu pak budou jednotlivé pixely daného obrázku. Směr vlnového vektoru  $\vec{k}_z$  tak bude dán jako rozdíl polohy pozorovatele a daného pixelu. Ze znalosti těchto dvou vektorů již lze jednoduše dopočítat kosinus jimi svíraného úhlu jako

$$\cos \theta_z = \frac{\vec{v}_p \cdot \vec{k}_z}{\|\vec{v}_p\| \|\vec{k}_z\|}. \quad (3.7)$$

Změna frekvence signálu však není jediným jevem, ke kterému dochází při relativistických rychlostech. Kdybychom dopočítali transformaci i pro rovnoběžnou složku vektrů  $\vec{k}$ , zjistili bychom, jak se liší úhel, pod kterým dopadá signál na přijímač. Dochází tak k úhlové deformaci obrazu, neboli aberaci, se kterou je spojená i změna jasu daného obrazu. Při pozorování trojrozměrných objektů je pak dokonce možné pozorovat jejich odvrácené strany jako důsledek konečné rychlosti šíření signálu. Ukázky pozorování počítačově modelovaných objektů jsou například v [7]. My se však dále zaměříme jen na barevné posunutí.

# Kapitola 4

## Struktura programu

Nyní se dostáváme k hlavním výsledkům této práce. Je zde představena funkce **Spektr**, která určí barevné spektrum pixelu a dále výsledný program, který přepočítává dopplerovké posunutí barev nahraných fotografií.

### 4.1 Pomocné funkce

Soubor `prfc.py` obsahuje pomocné funkce pro další práci.

Funkce `V(A,B)` a `VV(A,B)` vrátí normu vektoru  $B - A$  (dvousložkový a trojsložkový vektor).

Funkce `Vv(A,B,C)` vrátí vzdálenost bodu  $C$  od přímky dané body  $A$  a  $B$ .

Funkce `pro(A,B,C)` vrátí souřadnice kolmé projekce bodu  $C$  na přímku danou body  $A$  a  $B$  v rovině.

Funkce `inter(A,B,C,D)` vrátí souřadnice průsečíku přímek  $AB$  a  $CD$ .

Funkce `scale(V)` normuje souřadnice RGB, kde je nutné je převést z hodnot  $\langle 0, 255 \rangle$  na  $\langle 0, 1 \rangle$ .

### 4.2 Funkce Spektr

Funkce vrátí pole, které odpovídá barevnému spektru barvy RGB, která je na vstupu. Popis této funkce a následně i celého programu je zvolen jako přímý komentář kódu. Někdy bylo nutné rádek s příkazem rozdělit. Takové místo je vyznačeno na obou stranách jako "...".

```
import time
import numpy
import cv2
import imageio
import string

from prfc import Vv
from prfc import pro
from prfc import inter

def Spektr(R,G,B):
    #vstupem této funkce jsou RGB údaje o barvě pixelu
    #načtení hodnot pro vlnové délky z textového souboru,
    #ve které jsou uloženy informace o gamutu formát je
    #toto:
```

```

# vlnová délka      x      y
# 420          0.159581463    0.015892612

#délka souboru v řádcích
num_lines = sum(1 for line in open(r'Lambda.txt'))

file = open(r'Lambda.txt')

    #pole, ve kterém jsou údaje o okraji gamutu uloženy
Lambda = []
Min = 420 #maximální vlnová délka spektra v nm
Max = 703 #minimální vlnová délka spektra v nm

for i in range(num_lines):
    numbers = file.readline()
    numbers = numbers.split()
    for j in range(3):
        numbers[j] = float(numbers[j])

    Lambda.append(numbers)
file.close()

#načtení druhé sady spektra, pro které budeme hledat
#partnerky; již menší množství vlnových délek

num_lines = sum(1 for line in open(r'Spektrum.txt'))

file = open(r'Spektrum.txt')
    #pole, ve kterém jsou uloženy vlnové délky, pro které
#bude hledat partnerky
Spektrum = []
for i in range(num_lines):
    numbers = file.readline()
    numbers = numbers.split()
    for j in range(3):
        numbers[j] = float(numbers[j])

    Spektrum.append(numbers)
file.close()

Min2 = 420
Max2 = 700
Krok = 14      #vzdálenost vlnových délek spektrálních barev,
#které procházíme s maximální a minimální vlnovou délkou

#definování dL
#LS počet všech spektrálních barev kterým budou vybírány
#partnerky

```

```

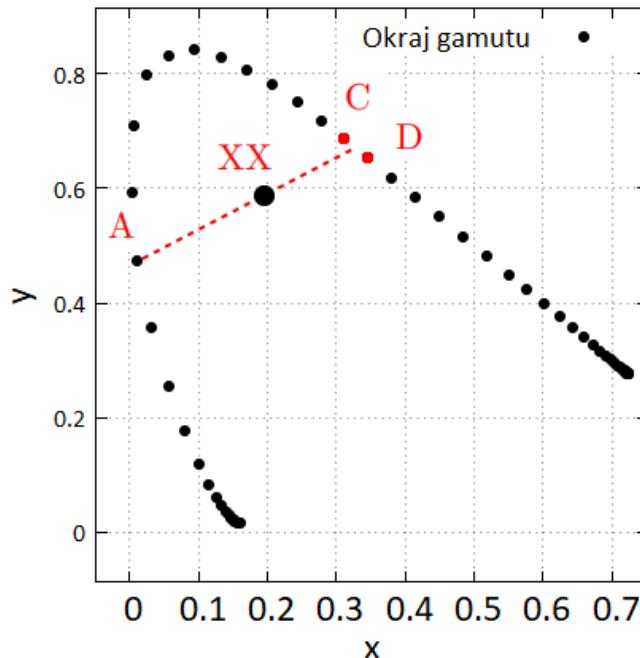
LS = len(Spektrum)

#počet všech spektrálních barev, které známe
#a ze kterých budou vybírány partnerky
L = len(Lambda)

#tady budou uloženy informace o barvách
#[barva A][0 = intenzita barvy A z pole Spektrum,
# 1 = pořadí barvy C v poli Lambda, 2 = intenzita barvy C,
#3 = intenzita barvy D]; barvy C a D jsou sousední
#spektrální barvy, jejichž smícháním dostaneme
#partnerku k barvě A
col = []
col = [[0 for i in range(4)] for j in range(LS+2)]

```

Nyní je vhodné podívat se znovu na to, jak je řešené hledání partnerské barvy. Orientačně je to znázorněno na obrázku 4.1. Pro barvu A hledáme průsečík přímky  $AXX$ , kde  $XX$  je barva, kterou chceme namíchat, s okrajem gamutu. Ten máme nyní rozdelený na jednotlivé barvy (jednotlivé body jsou uloženy v poli Lambda). Přímka tedy pravděpodobně neprotne okraj gamutu v jednom z uložených bodů barev v Lambda. Pravděpodobněji protne oblast mezi nějakými dvěma uloženými sousedními body  $C,D$ . Partnerská barva sice odpovídá skutečné spektrální barvě, tu však nemáme uloženou v našem poli Lambda a neznáme její vlnovou délku, proto k ní přistoupíme jako k ostatním barvám v gamutu a namícháme ji pomocí barev v bodech  $C$  a  $D$  (podle postupu míchání barev v gamutu zmíněném v kapitole 1). Mezi ně také rozdělíme intenzitu této namíchané partnerky.



Obrázek 4.1: Nalezení partnerky ke spektrální barvě v bodě A, tak že jejich kombinací je barva v bodě XX. Pokud partnerka není v gamutu definována, jsou jako partnerské barvy brány body C,D.

```

souc = int(R)+int(B)+int(G)
if souc > 0:
    R = scale(R)
    G = scale(G)
    B = scale(B)
    col[LS + 1][0] = 1
#převod na prostor XYZ
X = (0.4124564*R + 0.3575761*G + 0.1804375*B)
Y = (0.2126729*R + 0.7151522*G + 0.0721750*B)
Z = (0.0193339*R + 0.1191920*G + 0.9503041*B)

#převod na prostor xyY
x = X /(X + Y + Z)
y = Y /(X + Y + Z)

#define nování dL;
#proběhne LS hledání partnerek k LS spektrálních barev
lum = Y/LS

for i in range(LS):
    #načtení souřadnic spektrální barvy,
    #jijíž partnerku budeme hledat
    A = [Spektrum[i][1], Spektrum[i][2]]
    #souřadnice barvy, kterou chceme namíchat
    #v prostoru xyY
    XX = [x,y]

    for j in range(L - 2):
#najde barvu spektrální barvu A a začne od ní zkoušet,
#do kterého intervalu se trefí
        index = (i*Krok + j + 1)%L
        C = [Lambda[index][1], Lambda[index][2]]
#dvě sousední barvy a my vyzkoušíme, jestli jejich mix
#bude partnerkou pro A
        D = [Lambda[(index + 1)%L][1], Lambda[(index...
            ... + 1)%L][2]]

        P = inter(A,XX,C,D)#průsečík přímky AXX a CD

        if V(C,P) + V(D,P) - V(C,D) < 0.0000000001:
#pokud je průsečík na přímce CD, pak lze partnerku namíchat
#z barev CD a tedy je započteme

            K = A[1]*(P[0] - XX[0])/(P[1]*(XX[0] - A[0]))
#vztah 2.5

            col[i][0] = lum * K / (K + 1)
#vztah 2.5; intenzita první barvy A

```

```

        col[i][1] = index
#pozice barvy C; pozice D je automaticky index +1
        K = C[1]*(D[0] - P[0])/(D[1]*(P[0] - C[0]))
#vztah 2.5 pro mix C a D

        col[i][2] = (lum - col[i][0])* K / (K + 1)
#intenzita barvy C
        col[i][3] = lum - col[i][0] - col[i][2]
#intenzita barvy D

j = L - 2 #už není potřeba hledat

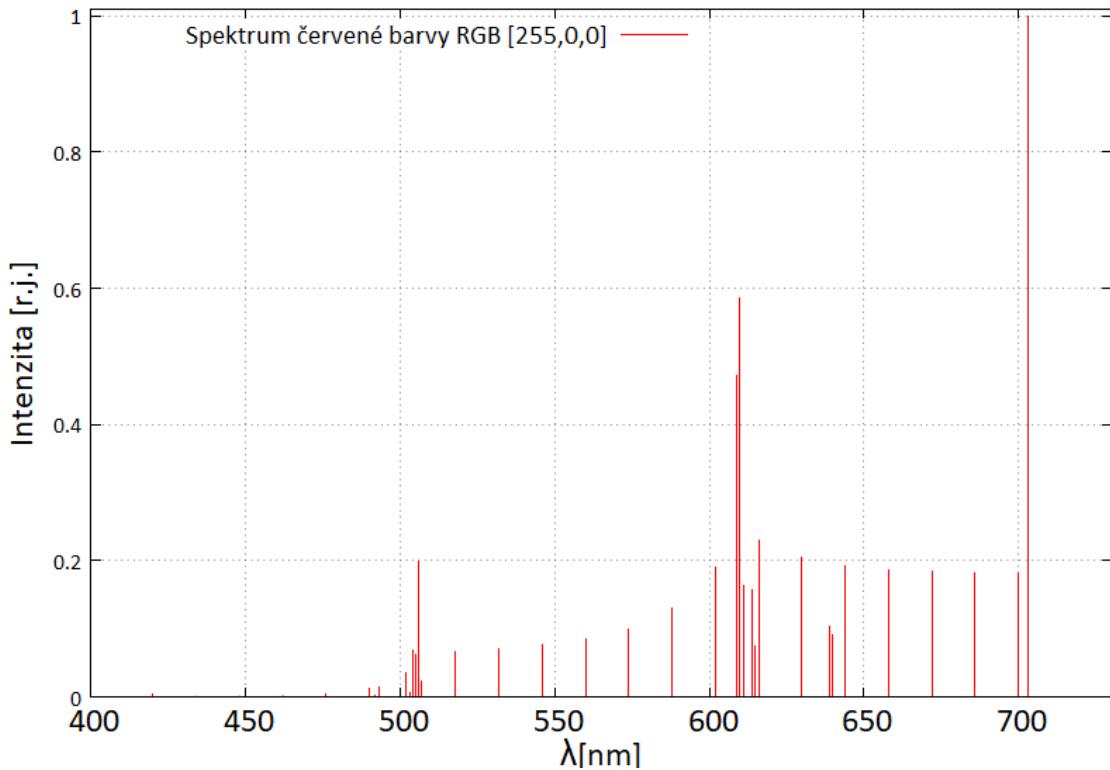
Spc = []
Spc = [[0 for i in range(2)] for j in range(L)]
for i in range(L):
    Spc[i][0] = Lambda[i][0]

for i in range(LS):
    Spc[int(i*Krok)][1] = Spc[int(i*Krok)][1] + col[i][0]
    Spc[col[i][1]][1] = Spc[col[i][1]][1] + col[i][2]
    Spc[(col[i][1]+1)\%L][1] = Spc[(col[i][1]+1)\%L][1]...
... + col[i][3]

return(Spc) #vrátí pole vlnových délek s intenzitou,
#kterou jsou ve spektru dané barvy se souřadnicemi RGB zastoupeny

```

Na obrázku 4.2 je znázorněno získané spektrum funkcí Spektr červené barvy s RGB souřadnicemi [255,0,0] (přesněji se jedná o souřadnice v sRGB). Jelikož červená barva [255,0,0] není spektrální barvou (tedy není na okraji gamutu, viz obr 1.5), není vyjádřena jen jednou vlnovou délkom.



Obrázek 4.2: Spektrum červené barvy s RGB souřadnicemi [255,0,0] v prostoru sRGB určené funkcí `Spektr`.

### 4.3 Program

Konečně jsme se dostali k popisu samotného programu. Uživatel zadá obrázek a vektor rychlosti pozorovatele. Na oplátku získá obrázek s dopplerovsky posunutými barvami dle vektoru rychlosti. Jelikož je fáze načtení obrázku časově náročná, lze zadat i vektor zrychlení, minimální a maximální rychlosť, které by měl pozorovatel dosáhnout a počet snímků, které mají být vykresleny. Na výstupu je tak sada obrázků s informací o příslušné rychlosti pozorovatele vůči původnímu obrazu. Uživatel však může udělat i další úpravy<sup>1</sup>, jako třeba nastavení počáteční polohy pozorovatele, nebo změnit jednotky, ve kterých je daný případ počítán. Více však už v následující sekci.

Listing 4.1: Python example

```
import time
import numpy
import cv2
import imageio
import string
import os
# importované funkce, které jsou popsány v sekci 4.1
from prfc import V
from prfc import Vv
from prfc import pro
```

<sup>1</sup>Jelikož se jedná jen kód bez uživatelského rozhraní, tak si může uživatel dělat samozřejmě, co chce.

```

from prfc import inter
from prfc import scale
from prfc import backscale
from prfc import VV

#Soustava souřadnic zdroje:
#Počátek [0,0,0] je v levém horním rohu obrázku (tady začíná
# Python obrázek načítat).
#Osa x je orientovaná směrem k dolnímu okraji obrázku. Osa y je
# orientovaná směrem
#k pravému okraji obrázku. Osa z je orientovaná kolmo na obrázek
# ve směru k pozorovateli.

#Jednotky jsou voleny tak, že c = 1; jednotky v rovině xy lze
# změnit konstantou pp na pixel/pp
Vmax = [0,0,-0.5]
Vmin = [0,0.05,0]
Start = [0.5,0.5,500] #Počáteční poloha pozorovatele; x,y
# souřadnice jsou násobky výšky a šířky obrázku; souřadnice z je
# pak vzdálenost od obrázku

pp = 1000

Accel = [0,0,-0.1] #Zrychlení, se kterým se mění rychlosť, pro
# kterou je obrázek přepočítán
#snahou není simulovat pohyb obecně zrychlujícího pozorovatele,
#ale jen napočítat dopplerovské posunutí pro více rychlostí

Frames = 100 #Počet snímků, které budou vytvořeny

time = (Vmax[2] - Vmin[2])/Accel[2] #Doba, po kterou se
#pozorovatel bude pohybovat je momentálně určena jen podle složky
# rychlosti v ose z
t=time/Frames #Doba, která uplyne mezi dvěma snímkami
jpg = '.jpg' #obrázky budou ukládány ve formátu jpg

images = []
Pic = r'Picture.jpg' #Adresa obrázku, se kterým se bude pracovat
img = cv2.imread(Pic) #Načtení původního obrázku

images = images + [img] #Zde se ukládají jednotlivé snímkы;
# z tohoto seznamu lze pak například vytvořit animaci přímo
# v Pythonu

```

```

#Tato fáze je popsaná v předchozí sekci o funkci Spectr

num_lines = sum(1 for line in open(r'Lambda.txt'))

file = open(r'Lambda.txt')

Lambda = []
Min = 420
Max = 703
for i in range(num_lines):
    numbers = file.readline()
    numbers = numbers.split()
    for j in range(3):
        numbers[j] = float(numbers[j])

    Lambda.append(numbers)
file.close()

num_lines = sum(1 for line in open(r'Spektrum.txt'))

file = open(r'Spektrum.txt')

Spektrum = []
for i in range(num_lines):
    numbers = file.readline()
    numbers = numbers.split()
    for j in range(3):
        numbers[j] = float(numbers[j])

    Spektrum.append(numbers)
file.close()

Min2 = 420
Max2 = 700
Krok = 14

LS = len(Spektrum)
L = len(Lambda)
col = []

height, width, channels = img.shape
stred = [height/2, width/2]

col=[[[[0 for x in range(4)] for ii in range(LS+2)] for j in...
     ... range(width)] for i in range(height)]

for i in range(height):
    for j in range(width):
        print(i,j) #Výpis pixelu, který je načítán

```

```

B = images[0][i,j,0]
G = images[0][i,j,1]
R = images[0][i,j,2]

souc = int(R)+int(B)+int(G)
if souc > 0:
    R = scale(R)
    G = scale(G)
    B = scale(B)

    col[i][j][LS + 1][0] = 1 #Pokud bude pixel černý,
# hodnota bude změněna na 0
    X = (0.4124564*R + 0.3575761*G + 0.1804375*B)
    Y = (0.2126729*R + 0.7151522*G + 0.0721750*B)
    Z = (0.0193339*R + 0.1191920*G + 0.9503041*B)

    x = X /(X + Y + Z)
    y = Y /(X + Y + Z)
    lum = Y/LS

    for ii in range(LS):
        A = [Spektrum[ii][1], Spektrum[ii][2]]
        XX = [x,y]

        for jj in range(L - 2):
            index = (ii*Krok + jj + 1)\%L
            C = [Lambda[index][1], Lambda[index][2]]
            D = [Lambda[(index + 1)\%L][1], Lambda[... ...
            ... (index + 1)\%L][2]]

            P = inter(A,XX,C,D)

            if V(C,P) + V(D,P) - V(C,D) < 0.0000000001:
                KK = A[1]*(P[0]-XX[0])/(P[1]*(XX[0]...
                ... - A[0]))
                col[i][j][ii][0] = lum * KK / (KK + 1)
                col[i][j][ii][1] = index
                KK = C[1]*(D[0] - P[0])/(D[1]*(P[0]...
                ... - C[0]))
                col[i][j][ii][2] = (lum - ...
                ... col[i][j][ii][0])* KK / (KK + 1)
                col[i][j][ii][3] = lum-col[i][j][ii][0]...
                ... - col[i][j][ii][2]

                jj = L - 2

```

```

    else:
        #Pixel byl černý
        col[i][j][LS + 1][0] = 0

#Zadání tří bodů, kterými je dán prostor sRGB; viz sekci 1.3
sR = [0.64 , 0.33]
sG = [0.30 , 0.60]
sB = [0.15 , 0.06]
#Vzdálenost bodů trojúhelníku, který definuje sRGB prostor
RG = V(sR,sG)
GB = V(sG,sB)
RB = V(sR,sB)
#Plocha trojúhelníku, který vymezuje sRGB prostor; bude
# zapotřebí, až budeme provádět projekce barev do tohoto prostoru
S_tr0 = RG * Vv(sR,sG,sB)/2

for w in range(Frames):
    #Výpočet složek rychlosti pro daný snímek
    Vx[0] = Vmin[0] + Accel[0]*w*t
    Vx[1] = Vmin[1] + Accel[1]*w*t
    Vx[2] = Vmin[2] + Accel[2]*w*t

    #Nastavení jména daného snímku
    name = r'Doppler_effect'
    name = name + str(w) + jpg

    for i in range(height):
        for j in range(width):

            Y = 0 #Intenzita výsledné barvy v prostoru xyY
            if col[i][j][LS + 1][0] == 1: #Pokud pixel nebyl
#černý

                L_citX = 0
                L_citY = 0
                L_jmen = 0
                #Výpočet poměru vlnových délek signálu
#reprezentovaných v soustavě pozorovatele P a zdroje Z
                Z =[i,j,0] #Poloha zdroje; jedná se o daný pixel
# na obrázku. Ten je dán jako rovina z = 0.

                #Nyní je dopočítána poloha pozorovatele
# v soustavě zdroje.
                P = [Start[0]*height +t*Vx[0]+Accel[0]*t**2...
/ 2,Start[1]*width + t*Vx[1] + Accel[1]*t**2 / 2,Start[2] +...
... t*Vx[2] + Accel[2]*t**2 / 2]
                #Výpočet vlnového vektoru v soustavě zdroje;
# je dán jako rozdíl poloh pozorovatele a zdroje

```

```

kv = [P[0] - Z[0], P[1] - Z[1], P[2] - Z[2]]
#Pokud je vektor rychlosti nebo vlnový vektor
# nulový, kosinus úhlu ve vztahu 3.6 je 0
if VV(Vx) == 0 or VV(kv) == 0:
    cost = 0
else:
    #Kosinus [hlu theta; vztah 3.7
    cost = (Vx[0]*kv[0]+Vx[1]*kv[1]...
... +Vx[2]*kv[2]) / (VV(Vx)*VV(kv))

        #Signál v našem případě není ve frekvenční
#reprezentaci, ale je vyjádřen pomocí vlnové délky. Jedná se
# o výraz 3.6
K = (numpy.sqrt(1 - VV(Vx)**2))/(1 - VV(Vx)*cost)

for ii in range(LS):
    #Dopplerovské posunutí vlnových délek
    Ind2 = int((col[i][j][ii][1] + Min)*K) - Min
#Vlnová délka, jejíž hodnota je uložena na pozici [i][j][ii][1]
# pole col je dopplerovksy posunuta
    Ind3 = int(((col[i][j][ii][1] + 1)\%L +...
... Min)*K) - Min #Posunutí vedlejší vlnové délky
    Ind = col[i][j][ii][1]

        #Pokud se posunuté vlnové délky nacházejí
# ve viditelném spektru, je proveden výpočet 1.1 smíchání barev
        if K * Spektrum[ii][0] >= Min2 and K *...
... Spektrum[ii][0] <= Max2:
            L_citX = L_citX+(Lambda[int((ii*Krok...
... + Min2)*K) - Min2][1]/Lambda[int((ii*Krok +Min2)*K)-...
... Min2][2])*col[i][j][ii][0] #Jsou započteny intenzity
# původních barev, ty však mají nyní nové souřadnice odpovídající
# posunutým barvám
            L_citY = L_citY + col[i][j][ii][0]
            L_jmen = L_jmen + ...
... col[i][j][ii][0]/Lambda[int((ii*Krok + Min2)*K) - Min2][2]
            Y = Y + col[i][j][ii][0]

        if K * Lambda[Ind][0] >= Min and K * ...
... Lambda[Ind][0] <= Max:
            L_citX = L_citX + ...
... (Lambda[Ind2][1]/Lambda[Ind2][2])*col[i][j][ii][2]
            L_citY = L_citY + col[i][j][ii][2]
            L_jmen = L_jmen + ...
... col[i][j][ii][2]/Lambda[Ind2][2]
            Y = Y + col[i][j][ii][2]

        if K * Lambda[(Ind + 1)\%L][0] >= Min ...
... and K * Lambda[(Ind + 1)\%L][0] <= Max:

```

```

        L_citX = L_citX + ...
... (Lambda[Ind3][1]/Lambda[Ind3][2])*col[i][j][ii][3]
        L_citY = L_citY + col[i][j][ii][3]
        L_jmen = L_jmen + ...
... col[i][j][ii][3]/Lambda[Ind3][2]
        Y = Y + col[i][j][ii][3]

        if L_jmen > 0:
            x = L_citX/L_jmen
            y = L_citY/L_jmen
        #Nyní máme smíchané celé spektrum a můžeme barvu
# převést do prostoru xyY
        if L_jmen > 0:
            #Jinak se jedná o černý pixel
            x = L_citX/L_jmen
            y = L_citY/L_jmen
            #Nyní známe polohu bodu v rovině xy; ještě
# musíme ověřit, že je i v prostoru sRGB
            XX = [x,y]

        #Nyní se musíme podívat na polohu naší barvy vůči
# trojúhelníku, který určuje prostor sRGB

        #Vzdálenost bodu XX od přímek, na kterých
# leží strany trojúhelníku sR sG sB
            XXrg = Vv(sR,sG,XX)
            XXgb = Vv(sG,sB,XX)
            XXrb = Vv(sR,sB,XX)
            #Vzdálenosti bodu XX od vrcholů trojúhelníku
            RXX = V(sR,XX)
            GXX = V(sG,XX)
            BXX = V(sB,XX)

            sXrg = pro(sR,sG,XX) #Kolmá projekce bodu XX
# na přímku danou body sR, sG
            sXgb = pro(sG,sB,XX) #Kolmá projekce bodu XX
# na přímku danou body sG, sB
            sXrb = pro(sR,sB,XX) #Kolmá projekce bodu XX
# na přímku danou body sR, sB

            #Vzdálenosti bodu XX a jeho projekcí na dané
# přímky
            sXXrg = V(XX,sXrg)
            sXXgb = V(XX,sXgb)
            sXXrb = V(XX,sXrb)

            if abs((V(sR,sXrg) + V(sG,sXrg)) - RG)>...
... 0.0000000001: #Pokud je argument roven nule, projekce bodu XX
#leží na úsečce sRsG, tedy na straně trojúhelníku; jinak jeho

```

```

#projekce neleží na této straně trojúhelníku a nemá smysl
#uvobožovat o této projekci

            sXXrg = sXXrg + sXXgb + sXXrb+RXX+GXX+BXX
            if abs((V(sG,sXgb) + V(sB,sXgb)) - GB)>...
... 0.0000000001:#Pokud je argument roven nule, projekce bodu
#XX leží na úsečce sBSG

            sXXgb = sXXrg + sXXgb + sXXrb+RXX+GXX+BXX
            if abs((V(sR,sXrb) + V(sB,sXrb)) - RB) >...
... 0.0000000001:#Pokud je argument roven nule, projekce bodu XX
# leží na úsečce sRsB

            sXXrb = sXXrg + sXXgb + sXXrb+RXX+GXX+BXX

            S_tr = 0.5 * (RG * XXrg + GB*XXgb + RB*XXrb)
#Součet ploch tří trojúhelníků sRsGXX, sGsBXX a sRsBXX;

            #Pokud je součet jejich obsahů roven obsahu
# trojúhelníku sRsGsB, pak bod XX v něm leží,
            #a tedy i v prostoru sRGB a není třeba jej
# projektovat

            if S_tr0 == S_tr or abs(S_tr0 - S_tr) <...
... 0.0000000001 :
            NotOK = 0*K #Obsahy se shodují a bod XX
# leží v prostoru sRGB

            else:
            #Bod XX leží mimo náš trojúhelník;
# na trojúhelníku se pokusíme najít bod, který je mu nejbližší

            if sXXrg == min(sXXrg, sXXgb, sXXrb, ...
... RXX, GXX, BXX):
                x = sXrg[0]
                y = sXrg[1]

            elif sXXgb == min(sXXgb, sXXrb, RXX, ...
... GXX, BXX):
                x = sXgb[0]
                y = sXgb[1]

            elif sXXrb == min(sXXrb, RXX, GXX, BXX):
                x = sXrb[0]
                y = sXrb[1]

            elif RXX == min(RXX, GXX, BXX):
                x = sR[0]
                y = sR[1]

```

```

        elif BXX == min(GXX,BXX):
            x = sB[0]
            y = sB[1]

        else:
            x = sG[0]
            y = sG[1]

    #Teď už můžeme převést barvu z xyY prostoru
# do XYZ a z něj do RGB
    XX = [x,y]

    X = Y*x / y
    Z = Y*(1 - x - y)/y

    R = 3.2404542*X - 1.5371385*Y - 0.4985314*Z
    G = -0.9692660*X + 1.8760108*Y + 0.0415560*Z
    B = 0.0556434*X - 0.2040259*Y + 1.0572252*Z

    #Normalizace
    R = backscale(R)
    G = backscale(G)
    B = backscale(B)

    else:
        R = 0
        G = 0
        B = 0

    #V průběhu výpočtu se mohly hodnoty v rámci strojové
# přesnosti dostat z intervalu [0,255], tak provedeme kontrolu
    else:
        R = 0
        G = 0
        B = 0
        if R < 0:
            R = 0
        if G < 0:
            G = 0
        if B < 0:
            B = 0

        if R > 255:
            G = int(G*255/R)
            B = int(B*255/R)
            R = 255
        if G > 255:
            R = int(R*255/G)
            B = int(B*255/G)
            G = 255

```

```

        if B > 255:
            G = int(G*255/B)
            R = int(R*255/B)
            B = 255

        images[w][i,j] = [B,G,R] #Uložení nových hodnot
# RGB pixelu

        #Do dostatečně velkých obrázků zapíšeme do spodního
# rohu rychlost, pro kterou byly dopočítány
        if height > 100 and width > 100:
            cv2.putText(images[w],str(Vx),...
... (int(width/20),height -int(height/20)), ...
...cv2.FONT_HERSHEY_SIMPLEX, 0.5,(0,200,200),2)
            cv2.putText(images[w],str(Vx), ...
... (int(width/20),height -int(height/20)), ...
...cv2.FONT_HERSHEY_SIMPLEX, 0.5,(0,0,0),1)

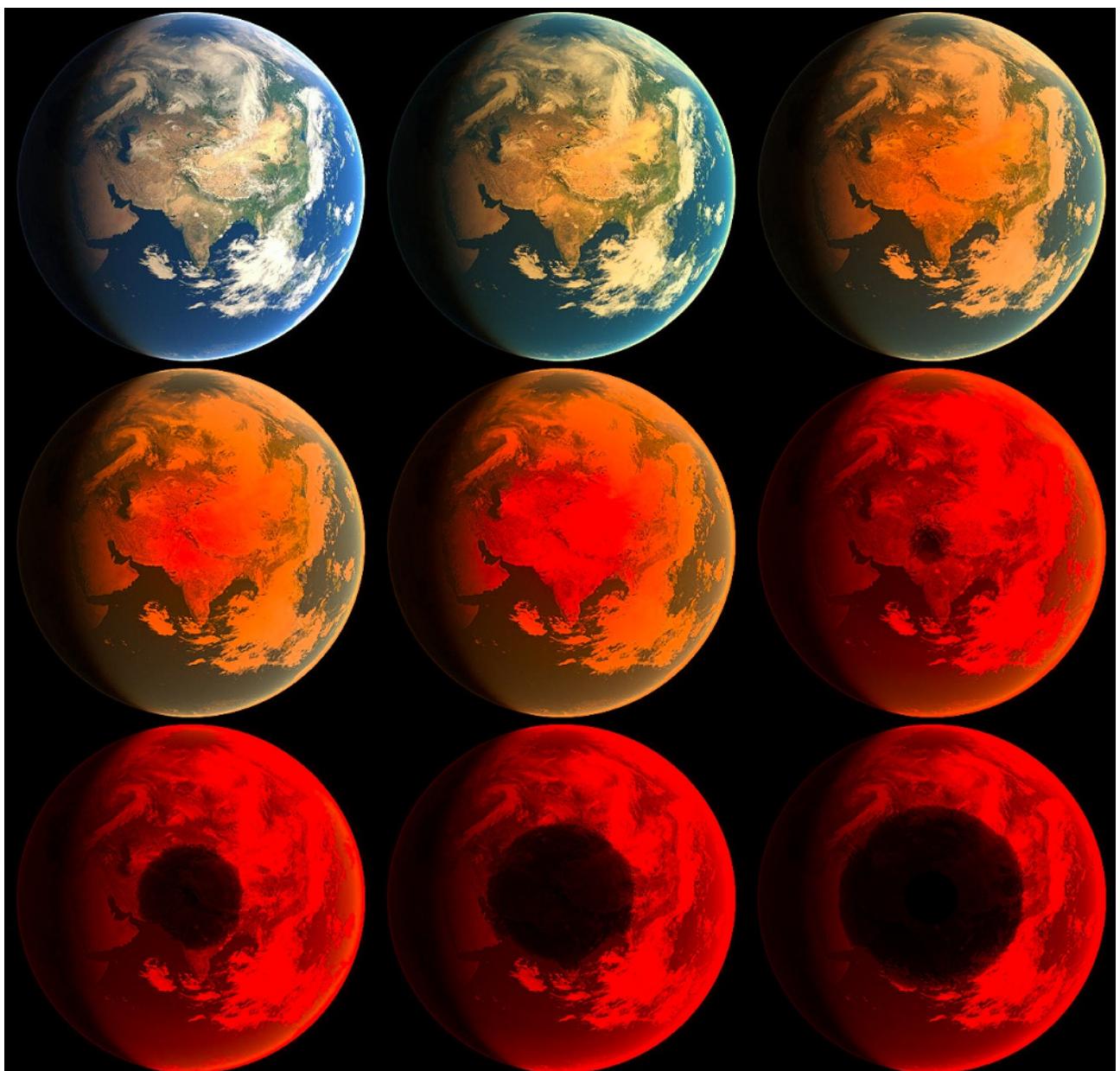
        #Vytvoření pracně vytvořeného obrázku s posunutými barvami
        cv2.imwrite(name,images[w])
        #Načtení nového obrázku, který bude přepočten pro novou
# rychlosť
        img = cv2.imread(Pic)
        images = images + [img]

print(r'konec') #To už je vše

```

## 4.4 Ukázky

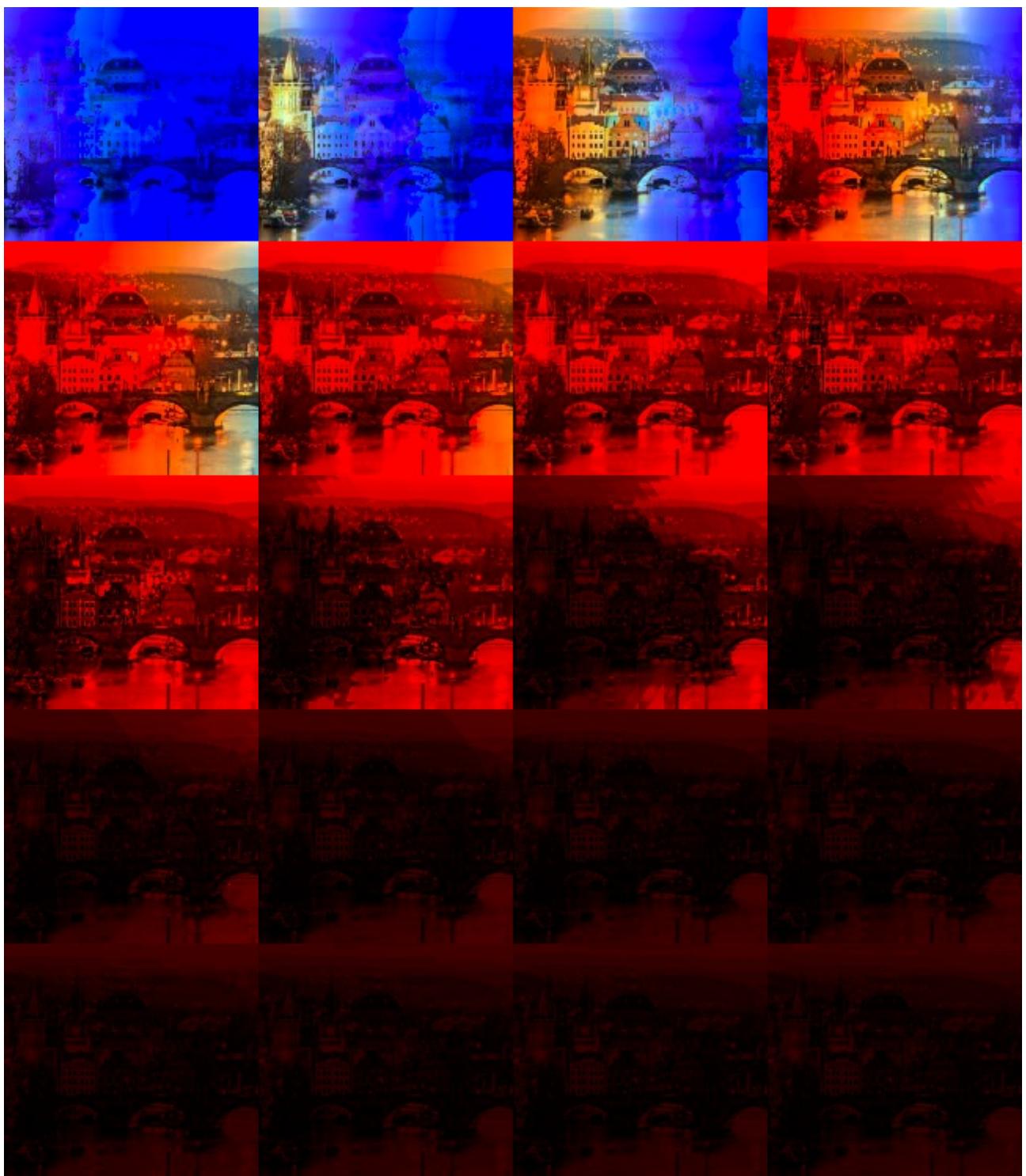
Nyní ukážeme několik příkladů zpracování obrázků. Také se podíváme na rozdíl mezi metodou, kterou jsme použili a případem, kdy jsou posouvány přímo RGB souřadnice jako spektrální barvy.



Obrázek 4.3: Vzdalování se od Země. Rychlosť roste lineárne z 0 na  $0,5c$ . Zdroj pôvodného obrázku [10]



Obrázek 4.4: Vzdalování se od Tima. Rychlosť roste lineárne z 0 na  $0,5c$ . Zdroj pôvodného obrázku [11]



Obrázek 4.5: Rychlý průlet kolem Prahy ve směru zprava doleva. Rychlosť je konstantná. Zdroj pôvodného obrázku [12]

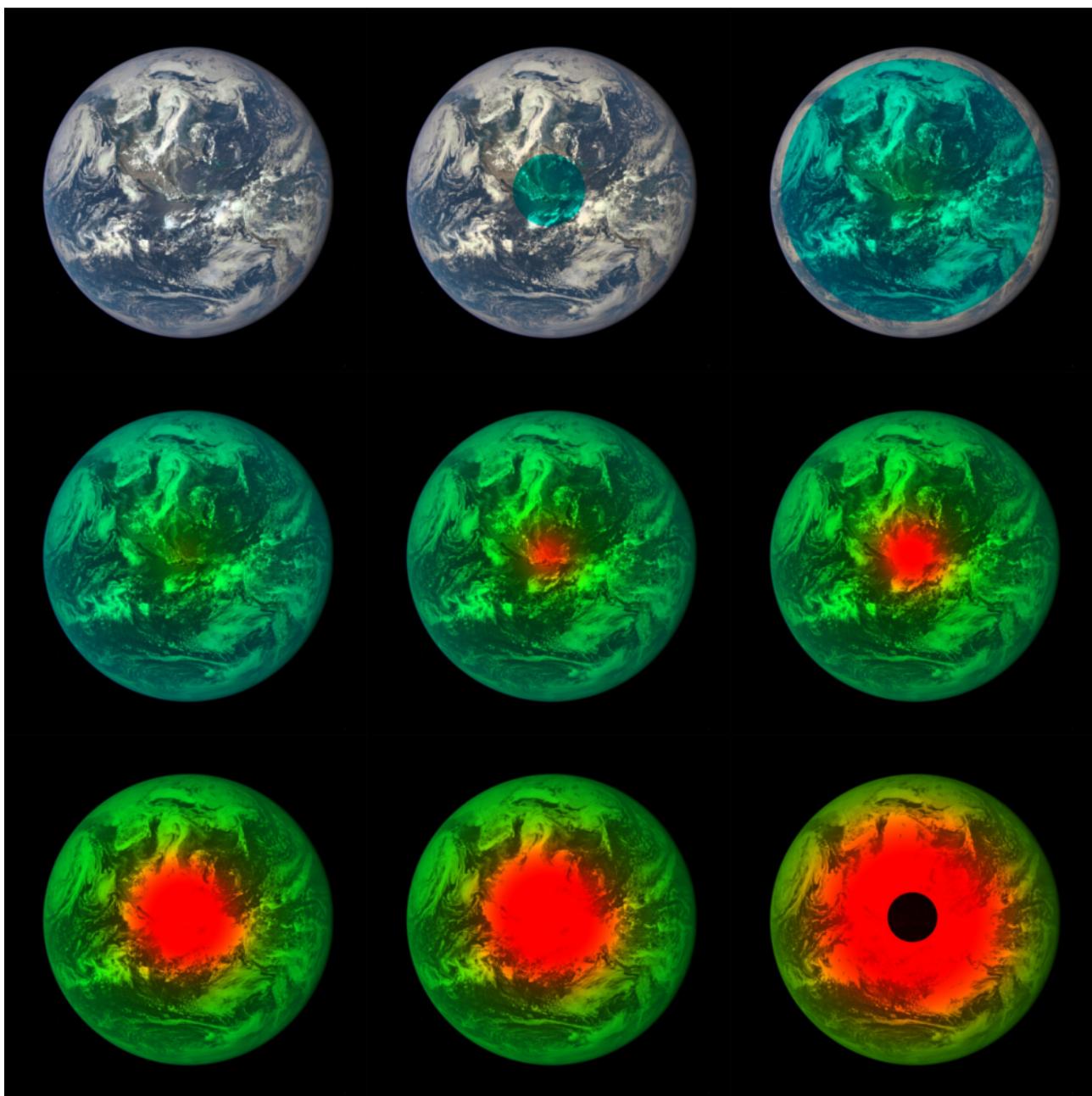


Obrázek 4.6: Rychlý průlet kolem Karlova s vypsaným vektorem rychlosti. Zdroj původního obrázku [13]

Hlavní výhodou vybudované metody, kterou jsme použili k přepočítání barev obrázků je, že k barevné změně dochází spojité v rámci možností. Postupně se mění nejen odstín barvy, ale následně také její intenzita s tím, jak se spektrum přesouvá do neviditelné oblasti. V tom se liší od případných jednodušších metod, kdy by byla barva reprezentována jednou hlavní spektrální barvou s doplnkovým barevným pozadím, případně pokud bychom posouvali přímo barevnou reprezentaci RGB a spektrum dané barvy by se skládalo pouze ze tří složek. U těchto metod by totiž docházelo ke skokovým změnám intenzity spojených s vystoupení složek spektra z viditelné oblasti. Námi zvolenou metodu lze chápat vlastně jako zjemnění dělení spektra. Spektrum získané metodou dominantní spektrální barvy by obsahovalo jednu dominantní složku spektra. Pouhý posun RGB složek, respektive jim přiřazeným spektrálním barvám, by vytvořil třísložkové spektrum. V naší metodě můžeme teoreticky volit libovolný nezáporný počet složek spektra, nicméně barvy ve zde uvedených obrázcích měly spektra složená z 20 až 40 spektrálních barev (nenastavuje se počet složek, ale jejich rozsah). Rozdíl můžeme porovnat na obrázku 4.7. Použitá metoda byla posun hodnot RGB. Spektra každé barvy byla tvorena stejnými třemi barvami a lišila se jen jejich intenzitou. Takovou podobu spekter různých barev bychom v přírodě, tedy po osvětlení bílým světlem, neočekávali. Tyto barvy byly dopplerovsky posunuty a převedeny zpět na RGB již stejným způsobem jaký popsán v sekci 2.4 a uveden v programu v sekci 4.3.

Při posunu barev obrázku 4.7 byla červená spektrální barva, která odpovídá složce R volena blízko okraje viditelného spektra. Navzdory vzdalování se od zdroje, kdy by se vlnová délka signálu měla prodlužovat, na druhém snímku vzniká ostrý modrý kruh. To je způsobeno právě posunem červené složky do neviditelné části spektra. Zelená i modrá složka se však posunují směrem k červené. Při rudém posunu tak snímek paradoxně skokově zmodral. Při pozorném pohledu najdeme i snímek, na které z viditelné oblasti přešla i druhá barevná složka, která byla původně zelená. Jedná se o snímky se sytě červenou středovou oblastí, ve které již nejsou barevně oddělené detaily, neboť spektrum těchto barev má jen jednu složku a tou je původně modrá složka, která byla posunuta do červené oblasti. Její skokový přechod do neviditelné oblasti je pak patrný na posledním snímku.

Je však nutné se zmínit, že se nejdená o ideálně zpracovanou metodu posunu RGB složek. Uvedená obrázek má sloužit pouze k zdůraznění nevýhod této metody oproti nám použitému postupu. Čtenář by měl také tendenci srovnávat tento obrázek s obrázkem 4.3, což samozřejmě může. Jedná se sice o stejnou planetu, ale její snímky jsou různé. U druhého byla také vzdálenost pozorovatele od zdroje, tedy snímku, volena výrazně nižší.



Obrázek 4.7: Vzdalování se od Země přepracované pouze posunem RGB složek spektra. Zdroj původního obrázku [14]

# Kapitola 5

## Závěr

Snahou bylo vytvořit program, který pozmění barvy vloženého obrázku podle relativistického Dopplerova jevu. Problematické bylo získání barevného spektra barev jednotlivých pixelů obrázku, neboť dostupné metody neposkytovaly spektra s reálným charakterem. Proto byla vytvořena nová metoda, díky které je možné reprezentovat libovolnou barvu pixelu teoreticky spojitym barevným spektrem. V jazyce Python byla napsána funkce **Spektr**, tak aby bylo nyná možné tuto metodu jednoduše implementovat i do jiných počítačových kódů. Její podstata je zde podrobně vysvětlena, proto může být případný zájemce o její použití schopný tuto funkci přepsat i do jiného programovacího jazyka.

Konečně byl napsán i toužený program, který dopplerovsky posunuje barvy. Uživatel může vložit obrázek s vektorem rychlosti, kterou se pozorovatel k obrázků blíží a na výstupu získá snímek s posunutými barvami. Díky nové použité metodě jsou barevné změny spojité, což odpovídá i jejich předpokládanému reálnému chování. I tak se výsledky od reálné situace liší, neboť zatímco v našem případě při posunu spektra z viditelné oblasti dochází k tmavnutí snímku, v reálném případě by se do viditelné oblasti dostaly složky, které přísluší ultrafialové nebo infračervené oblasti. Informace o této části spektra není možné z fotografie získat. Kód programu je sestaven a podrobně popsán tak, aby případné modifikace byly pro uživatele co nejjednodušší.

Zde uvedený kód je k dispozici na adrese

<https://github.com/JakNov/Light-spectrum>

# Literatura

- [1] CH. ABRAHAM, A Beginner's Guide to (CIE) Colorimetry, *internet*, citováno 26.4.2019 z:  
<https://medium.com/hipster-color-science/a-beginners-guide-to-colorimetry-401f1830b65a>
- [2] WIKIPEDIA, Gamut 1931 color space, *internet*, citováno 26.4.2019 z:  
[https://en.wikipedia.org/wiki/Gamut#/media/File:CIE1931xy\\_gamut\\_comparison.svg](https://en.wikipedia.org/wiki/Gamut#/media/File:CIE1931xy_gamut_comparison.svg)  
[https://upload.wikimedia.org/wikipedia/commons/thumb/3/3b/CIE1931xy\\_blank.svg/450px-CIE1931xy\\_blank.svg.png](https://upload.wikimedia.org/wikipedia/commons/thumb/3/3b/CIE1931xy_blank.svg/450px-CIE1931xy_blank.svg.png)
- [3] WIKIPEDIA, CIE 1931 color space, *internet*, citováno 26.4.2019 z:  
[https://en.wikipedia.org/wiki/CIE\\_1931\\_color\\_space](https://en.wikipedia.org/wiki/CIE_1931_color_space)
- [4] RIT, Useful Color Data, *internet*, citováno 26.4.2019 z  
[https://www.rit.edu/cos/colorscience/rc\\_useful\\_data.php](https://www.rit.edu/cos/colorscience/rc_useful_data.php)
- [5] G. WYSZECKI, W.S. STILES, Color Science (John Wiley & Sons, New York, 1982)
- [6] W. RINDLER, Relativity: Special, General, and Cosmological (Oxford University Press, 2001)
- [7] M. CHANG, F. LAI AND W. CHEN, Image Shading Taking into Account Relativistic Effects, ACM Transactions on Graphics, Vol. 15, No. 4, 1996.
- [8] D. WEISKOPF, Visualization of Four-Dimensional Spacetimes (Dizertační práce, Tübingen, 2001)
- [9] P.J.E. PEEBLES, D.T. WILKINSON, Physical Review 174, 2168 (1968)
- [10] NASA, Earth image, *internet*, citováno 26.4.2019 z:  
[http://www.nasa.gov/centers/goddard/images/content/638831main\\_globe\\_east\\_2047.jpg](http://www.nasa.gov/centers/goddard/images/content/638831main_globe_east_2047.jpg)
- [11] J.M. WARGO, Tim the Enchanter, *internet*, citováno 26.4.2019 z:  
<https://johnwargo.com/images/stories/2017/tim-the-enchanter-small.jpg>
- [12] ZLAVOMAT, Praha, *internet*, citováno 26.4.2019 z:  
<https://zlavomat.sgcndn.cz/images/t/944x472c/37/54/3754946-2aff43.webp>
- [13] MFF, Karlov, *internet*, citováno 26.4.2019 z:  
<https://www.mff.cuni.cz/vnitro/vvv/karlov.jpg>
- [14] NASA, Earth image, *internet*, citováno 26.4.2019 z:  
[http://www.nasa.gov/sites/default/files-thumbnails/image/187\\_1003705\\_americas\\_dxm.png](http://www.nasa.gov/sites/default/files-thumbnails/image/187_1003705_americas_dxm.png)