Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Společná laboratoř optiky UP a FZÚ AV ČR

BAKALÁŘSKÁ PRÁCE

Analýza obrazu v programu Wolfram Mathematica a její využití v praxi

Autor:	Zuzana Svozilíková
Studijní program:	B1701 Fyzika
Studijní obror:	1701R030 Přístrojová fyzika
Forma studia:	Prezenční
Vedoucí práce:	Mgr. Jan Říha, Ph.D.
Termín odevzdání práce:	Srpen 2019

Prohlašuji, že jsem předloženou bakalářskou práci vypracovala samostatně pod vedením Mgr. Jana Říhy, Ph.D. a že jsem použila zdrojů, které cituji a uvádím v seznamu použitých pramenů.

V Olomouci ……. ...…….....

Bibliografická identifikace:

Jméno a příjmení autora	Zuzana Svozilíková
Název práce	Analýza obrazu v programu Wolfram Mathematica a její využití v praxi
Typ práce	Bakalářská
Pracoviště	Společná laboratoř optiky
Vedoucí práce	Mgr. Jan Říha, Ph.D.
Rok obhajoby práce	2019
Abstrakt	Tato práce se zaměřuje na funkce pro zpracování 2D obrazu v programu Wolfram Mathematica. Součástí práce je přehled a popis funkcí. Následné využití funkcí v praxi bylo provedeno na proměnné hvězdě a analýze fotografie.
Klíčová slova	Wolfram Mathematica, analýza obrazu, proměnná hvězda.
Počet stran	74
Počet příloh	1 CD-ROM
Jazyk	Český

Bibliographical identification:

Author's first name and surname	Zuzana Svozilíková
Title	Image processing and analysis using Wolfram Mathematica and its practical implementation
Type of thesis	Bachelor
Department	Joint Laboratory of Optics
Supervisor	Mgr. Jan Říha, Ph.D.
The year of presentation	2019
Abstract	This thesis focuses on 2D image processing functions in Wolfram Mathematica. The work is an overview of these functions with their corresponding descriptions. Subsequently, practical uses of functions were presented on a variable star and photograph analysis.
Keywords	Wolfram Mathematica, image analysis, variable star.
Number of pages	74
Number of appendices	1 CD-ROM
Language	Czech

Ráda bych poděkovala svému vedoucímu práce Mgr. Janu Říhovi, Ph.D. za odborné vedení, ochotu a cenné rady, které mi poskytl při zpracování této bakalářské práce.

Dále bych poděkovala Mgr. Davidu Smrčkovi za cenné rady a pomoc při zpracování bakalářské práce v programu Wolfram Mathematica.

Též bych ráda poděkovala své rodině a přátelům, kteří mi byli oporou v průběhu studia a při jeho zakončení.

Obsah

Úvod

Wolfram Mathematica patří v současnosti mezi nejuznávanější počítačové, webové a cloudové společnosti na světě. Ačkoliv se využívá především pro vědecké, technické a matematické výpočty, Wolfram Language obsahuje funkce i na zpracování 2D obrazu.

První část práce se zaměřuje na historii, strukturu a principy programování v programu Wolfram Mathematica. Následně zde bude popsána analýza obrazu, datové formáty pro ukládání obrazových souborů, barva a charakteristiky obrazu.

Dále bude uveden přehled a popis funkcí pro zpracování 2D obrazu, které Wolfram Mathematica obsahuje.

Poslední část práce je soustředěna na využití funkcí v praxi. Bude zkoumána proměnná hvězda, změna jejího poloměru a magnitudy v čase. Dále bude provedena analýza zaměřená na analýzu osob na fotografii.

Jelikož Wolfram Mathematica umožňuje i textový zápis, bude tato práce vypracována přímo v programu Wolfram Mathematica.

1 Wolfram Mathematica

1.1 Historie

Společnost Wolfram Research založil v roce 1987 Stephan Wolfram. Stephan Wolfram (narozen 29. 8. 1959, Londýn) je britsko-americký matematik a fyzik. V letech 1979-1981 vedl vývoj počítačového systému pro algebru SMP (Symbolic Manipulation Program), což byl předchůdce programu Mathematica, ale z důvodu sporů s fakultou ohledně autorských práv odešel z Caltechu [1].

První kód programu Mathematica byl napsán v říjnu roku 1986. Stephan Wolfram se stal generálním ředitelem Wolfram Research. Jeho původním plánem bylo společnost soustředit na výzkum, vývoj a distribuci Mathematicy především prostřednictvím výrobců počítačů. Jako první smlouvu uzavřel se Steve Jobsem, který Mathematicu vložil do všech nevydaných počítačů NeXT. První verze programu Mathematica 1.0 byla zveřejněna 23. 6. 1988 [2].

Od vydání bylo celkově zveřejněno 12 verzí programu, poslední verze 12 byla zveřejněna 16. 4. 2019. Přehled verzí a jejich uvolnění v tabulce 1 [3].

Tabulka 1: Přehled verzí Wolfram Mathematica a jejich uvolnění. Převzato a upraveno z [3].

Verze	Datum uvolnění	Doba od posledního vydání v letech
1	Červen 1988	–
2	Leden 1991	2, 5
3	Září 1996	5, 8
4	Květen 1999	2, 8
5	Červen 2003	3
6	Květen 2007	4
7	Listopad 2008	1, 3
8	Listopad 2010	2
9	Listopad 2012	2
10	Červenec 2014	1, 8
11	Srpen 2016	2, 1
12	Duben 2019	2, 75

V roce 2009 byla spuštěna Wolfram Alpha, vytvořená na základě Wolfram Language, která využívá rozsáhlé výpočetní znalosti. Snaží se na rozdíl od vyhledávajících služeb, které poskytnou seznam webových stránek, odpovědět přímo na otázku zadanou uživatelem [4].

1.2 Struktura

Program Mathematica lze rozdělit do tří částí: Kernel, Front End a Packages.

- ***Kernel*** – jedná se o výpočtové jádro programu, provádí veškeré výpočetní operace.
- ***Front End*** – vlastní uživatelské prostředí. Zajišťuje komunikaci Kernelu s uživatelem. Základem je Notebook, do kterého uživatel zapisuje příkazy a je rozdělený na výpočetní buňky. Dále umožňuje zápis formátovaného textu.
- ***Packages*** – jedná se o doplňující systémové knihovny [5].

1.3 Programování

Wolfram Language je programovací jazyk vytvořený společností Wolfram Research, který využívá Wolfram Mathematica a Wolfram Programming Cloud. Mezi hlavní principy, které Wolfram Language odlišují od ostatních programovacích jazyků jsou např. vestavěná databáze, automatizace ve formě meta-algoritmů a superfunkcí, vestavěné porozumění přirozenému jazyku či reprezentace jako symbolický výraz [6].

Ve Wolfram Mathematice můžeme narazit na několik typů programování.

● ***Jednoduché programování*** – mezi toto programování můžeme zařadit nejběžnější datové typy, a to práce s čísly, řetězci, symboly a tzv. Lists, což jsou speciální datové struktury.

● ***Procedurální programování*** – tento styl programování je známý především z programovacích jazyků jako Fortran nebo C. Je založené na strukturách (např. For, While, If). Největší výhodou tohoto programování je, že pomocí struktur lze dlouhý program přepsat do několika řádků kódu.

● ***Funkcionální programování*** – se využívá při manipulaci se seznamy (Lists) nebo iteračními funkcemi, kde aplikujeme funkce vytvořené uživatelem na argumenty.

● ***Rekurzivní programování*** – při tomto programování lze rozlišit 2 metody, použití rekurzivních funkcí a použití rekurzivních transformací. Rekurzivní funkce aplikují globální pravidla, kdežto rekurzivní transformace lokální pravidla [7].

● ***Programování rule-based*** – programování je založeno na pravidlech. Příklady uvedeny v tabulce 2.

Tabulka 2: Příklad pravidel u programování rule-base. Převzato a upraveno z [7].

Operace	Význam	Příklad
=	Přiřazení hodnoty	`x = 3`
:=	Definice funkce	`f[x_] := x Sin[x]`
→	Provedení transformace	`a x + b /. x → 3`

Všechny tyto operace můžeme považovat za pravidla, například:

`x = 3` – jestliže se někde objeví `x`, bude mu přiřazena hodnota 3;

`f[x_] := x Sin[x]` – kdykoliv se objeví funkce `f[x_]`, bude nahrazeno x `Sin[x]`;

`a x + b /. x -> 3` – když se ve výrazu objeví `x`, bude nahrazeno hodnotou 3 [7].

2 Analýza obrazu

Pro ukládání obrazových dat využíváme datové formáty, které můžeme rozdělit na nekomprimované a komprimované.

● ***Nekomprimované formáty*** – jsou poměrně velké a jednomu pixelu odpovídá jeden datový úsek.

● ***Komprimované formáty*** – mají menší velikost a komprimace může být bezztrátová nebo ztrátová [8].

Dále můžeme rozdělit grafické formáty podle typu na rastrové nebo vektorové.

● ***Rastrové*** – ukládají se jako sekvence obrazových bodů a výsledný obraz vznikne jejich složením. Jejich největší nevýhodou je velikost, což vedlo ke vzniku formátů podporující kompresi dat, která může být ztrátová nebo bezztrátová. Při kompresi dochází ke ztrátě informace a lze nastavit kompresní poměr.

● ***Vektorové*** – reprezentují obraz jako množinu křivek a parametrů těchto křivek. Objekty jsou reprezentovány jako matematické funkce. Jejich výhoda je v tom, že při několikanásobném zvětšení nedochází k rasterizaci obrazu [8], [9].

Kompresi u rastrového obrazu dělíme na bezztrátovou a ztrátovou.

● ***Bezztrátová*** – nedochází ke ztrátě informace při uložení dat. Je založena na opakování hodnot jednotlivých barevných pixelů. Následně vhodným matematickým postupem se sníží počet čísel nutných k reprezentaci informací v obraze.

● ***Ztrátová*** – dochází ke ztrátě informace při uložení dat. Tuto ztrátu lidské oko nevnímá nebo vnímá velmi slabě, a tak celkový vizuální vjem by neměl být ovlivněn [8].

Mezi nejpoužívanější datové formáty řadí BMP, GIF, PNG, TIFF, Raw a JPEG. Přehled nejběžnějších rastrových formátů je uveden v tabulce 3.

Tabulka 3: Přehled nejběžnějších rastrových formátů. Převzato a upraveno z [8], [10].

Přípona	Komprese	Typ komprese	Barvy	Barevná hloubka $[\text{bit}/\text{px}]$
BMP	–	Bez komprese, RLE	Model RGB	1, 4, 8, 24
GIF	Bezztrátová	LZW	Paleta	1 – 8
PNG	Bezztrátová	LZ77, Huffmanovo	Model RGB	1 – 48
JPG	Ztrátová	DCT	Model $YC_B C_R$	max. 24
TIFF	Volitelná	Bez komprese, RLE, LZW	Volitelné	1 – 24

2.1 Datové formáty

2.1.1 BMP (Microsoft Windows Bit Mapped Picture)

Jedná se o rastrový formát, který pracuje s 24 bitovou hloubkou. U menší barevné hloubky může být využita komprese, ovšem obvykle se používá nekomprimovaná forma. Je to velmi rozšířený formát, ovšem má velkou datovou velikost, a proto se v současnosti využívají především formáty PNG či TIFF [8], [10].

2.1.2 GIF (Graphics Interchange Format)

Jde o rastrový komprimovaný formát s bezztrátovou kompresí, který však dokáže uložit pouze 256 barev. Jeden pixel reprezentuje jeden bit, ovšem jedná se pouze o index v tabulce definovaných barev. Tento formát umožňuje nastavit průhledné pozadí, nabízí možnost uložení více obrázků v jednom souboru či možnost animace [10], [11].

2.1.3 PNG (Portable Network Graphic Format)

Tento rastrový komprimovaný formát s bezztrátovou kompresí umožňuje uložení obrazu až ve 48 bitové barevné hloubce. Využívá barevný model RGBA, kde „A“ je alfa kanál, který nese informaci o průhlednosti. Využívá se jako náhrada formátu GIF při zobrazení grafických prvků v internetovém prohlížeči [8], [10].

2.1.4 TIFF (Tag Image File Format)

Jedná se o velmi univerzální formát, který umožňuje uložení obrazu ve vysoké kvalitě. Často se využívá pro fotografie k tisku. Jedná se o bezztrátový formát s 24 bitovou barevnou hloubkou. Jeho nevýhodou je velká datová velikost a jeho nemožnost zobrazení v internetovém prohlížeči, proto je převáděn do formátu JPEG [10], [11].

2.1.5 RAW

Formát RAW se využívá při ukládání digitálních fotografií, ukládání dat přímo z CCD snímače. Jde o nejkvalitnější formát. Tento formát je bezztrátový a nekomprimovaný, ovšem po úpravě fotografie je nutné uložit snímek v jiném formátu. Ukládá se zvlášť barevná a jasová složka. Umožňuje téměř neomezené možnosti úprav – např. vyvážení bílé, úprava podexponovaných míst snímku či lepší korekci expozice [10], [11].

2.1.6 JPEG (Joint Photographic Expert Group File Format)

Jde o nejrozšířenější formát, který je ztrátový, tudíž výsledná kvalita obrazu závisí na míře komprese. Je vhodný pro obraz s velkým počtem barev (ukládá ve 24 bitové barevné hloubce) a barevných přechodů, ovšem je nevhodný pro obraz s velkými jednobarevnými plochami a ostrými hranami, při kterém se uplatňuje Gibbsův jev [8], [10], [11].

2.2 Barva

Jedná se o kvalitu záření a souvisí se spektrálními vlastnosti záření. Barvu vnímáme díky sítnici oka, kde se vytváří vjem elektromagnetického záření. Na sítnici oka se nacházejí populace čípků, které jsou citlivé na tři základní barvy: červenou, zelenou a modrou. V tabulce 4 uveden přehled vlnových délek barev [8].

Tabulka 4: Přehled vlnových délek barev. Převzato a upraveno z [12].

Barva	Rozsah vlnové délky [nm]
Červená	625 – 800
Oranžová	590 – 625
Žlutá	565 – 590
Zelená	520 – 565
Azurová	500 – 520
Modrá	430 – 500
Fialová	400 – 430

Obraz můžeme rozdělit na černobílý nebo barevný.

- ***Černobílý obraz (šedotónový)*** – k prezentaci jasu stačí pouze jediné číslo, kde 0 označuje černou, naopak nejvyšší číslo bílou barvu. Výhoda černobílého obrazu je jednoduchost při zpracování algoritmy či analýzou obrazu a je výstupem většiny zobrazovacích metod.
- ***Barevný obraz*** – kromě informace o jasu obsahuje i informaci o barevných vlastnostech obrazu. Barevný model popisuje barvy na základě podílu jednotlivých složek. Míchání barev může

být buď aditivní nebo subtraktivní.

● ***Aditivní míchání*** – jednotlivé barevné složky se sčítají a výsledkem je světlo s větší intenzitou jasu. Základní barvy jsou červená (Red), zelená (Green) a modrá (Blue). Smícháním vznikne bílá barva.

● ***Subtraktivní míchání*** – s každou přidanou barevnou složkou se ubírá část původního světla. Základní barvy jsou azurová (Cyan), purpurová (Magenta) a žlutá (Yellow). Smícháním všech těchto barev vznikne černá barva [8].

Mezi základní barevné modely patří RGB a CMY. RGB barevný model vychází z aditivního míchání barev a je založen na lidském vnímáním barev, kdy složky odpovídají citlivosti čípků lidského oka. Největší využití tohoto barevného modelu je v obrazovkách. Ze subtraktivního míchání vychází CMY model. CMY model je často doplňován o „K" barvu (CMYK), který obsahuje čtvrtou barvu (Key black), a to z důvodu, že smíchaná černá z CMY není příliš kvalitní a přidáním "K" barvy lze snížit náklady na tisk [8], [13].

Dalšími barevnými modely jsou HSV (Hue, Sutaration, Value) či HSB (Hue, Sutaration, Balance) se základními parametry - tón, odstín a jas. Tyto barevné modely jsou založeny na tom, že při vnímání barev posuzujeme tři kvality a z nich modely vycházejí: barevný tón, sytost barvy a jas. Tedy jak se barevná kvalita zpracovává naším vědomím. Vylepšeným HSV je barevný model HSL (Hue, Saturation, Lightness) a model YUV složkou Y popisuje jas a UV chrominancí barevnou složku [8], [13].

2.3 Charakteristiky obrazu

Mezi globální charakteristiky se řadí hloubka obrazu, dynamika rozsahu, jas, kontrast a histogram. Těmito charakteristikami popisujeme obraz jako celek.

2.3.1 Hloubka obrazu (bitová hloubka)

Hloubka obrazu znamená počet bitů použitých pro uložení barvy [14]. U černobílého obrazu stačí 1 bit – rozlišujeme stavy černá nebo bílá. Naopak u barevného obrazu je nutná informace o třech barevných kanálech. Dle tabulky 5 vidíme, že více bitů na pixel znamená větší barevnou škálu, ale také vyšší paměťovou náročnost pro uložení grafické informace [15].

Tabulka 5: Přehled používaných barevných hloubek. Převzato a upraveno z [16].

Bitová hloubka [bit]	Počet barev	Označení
1	2	Mono Color
4	16	
8	256	
15	32 768	Low Color
16	65 536	High Color
24	16 777 216	True Color
32	4 294 967 296	Super True Color
48	281 474 976 710 656	Deep Color

2.3.2 Dynamický rozsah

Dynamický rozsah doplňuje hloubku obrazu, charakterizuje skutečné zobrazení jasových poměrů. Dynamický rozsah můžeme vyjádřit poměrem jasu nejtmavšího pixelu $a_{\min}$ a jasu nejsvětlejšího pixelu $a_{\max}$ jako

$$DRR = a_{\max} : a_{\min}, \tag{1}$$

kdy výsledný tvar obvykle upravujeme do tvaru N:1. Dále dynamický rozsah můžeme vyjádřit logaritmem a to jako

$$DR_L = 20 \log \frac{a_{\max}}{a_{\min}}. \quad (2)$$

Výsledek bude v jednotkách dB [8].

Dynamický rozsah můžeme rozdělit na dynamický rozsah scény a dynamický rozsah senzoru ve fotoaparátu.

- ***Dynamický rozsah scény (dynamic range)*** – udává poměr mezi nejsvětlejším a nejtmavším pixelem. Záleží pouze na rozpětí světel a stínů ve scéně, nikoliv na celkové světlosti scény.
- ***Dynamický rozsah senzoru*** – senzor je vybavený světlocitlivými buňkami. Pokud dopadne malé množství světla, nevybudí se žádný signál. Jestliže dopadne více světla na senzor než je schopna světlocitlivná buňka zpracovat, vygeneruje pouze maximální signál, ne větší. Tomuto rozsahu říkáme dynamický rozsah senzoru, ve kterém každá buňka senzoru pracuje dobře. Pokud bude fotocitlivá buňka mimo obraz, nebude dobře pracovat, čili nebude měřit světlo. Pomocí expozičního času a clony můžeme upravit, aby se na fotocitlivé buňky dostalo přesně tolik světla, aby se dynamický rozsah scény vešel do dynamického rozsahu senzoru [17], [18].

2.3.3 Jas

Jas obrazu je střední jas všech pixelů v obraze. Jedná se o subjektivní fotometrickou veličinu, tedy velkou roli ve vnímání hraje rozdílná citlivost lidského oka na jednotlivé barvy. Např. lidské oko je mnohem citlivější na žlutou barvu než na fialovou, proto se žlutý předmět bude zdát být jasnějším než fialový. Pokud máme barevný model RGB, vyjádříme jas barevného obrazu pomocí empirického vztahu

$$I = 0,299\,R + 0,587\,G + 0,114\,B, \quad (3)$$

kde *R, G, B* označují jednotlivé barevné složky [8], [19].

2.3.4 Kontrast

Kontrast vyjadřuje rozdíl mezi nejsvětlejším a nejtmavším pixelem obrazu. Jestliže kontrast zvyšujeme nebo snižujeme, způsobujeme změnu barevné škály a vnímání obrazu. Dále úpravou kontrastu můžeme detailněji rozpoznat objekty [20].

Nasvícení scény můžeme dvojím způsobem:

- ***Low key*** (tmavá tonalita, nízký klíč, nízká tónina) – převaha tmavých tónů a vysokého kontrastu mezi světlem a stínem. Poměr jasu světel a stínů dosahuje až 8:1.
- ***High key*** (světlá tonalita, vysoký klíč, vysoká tónina) – převaha světlých tónů a malý kontrast mezi světlem a stínem. Poměr jasu světel a stínů se může přiblížit až 1:1 [21].

2.3.5 Histogram

Histogram stupňů šedi je základní zdroj, který zobrazuje poměry jasu a kontrastu v obraze. Určuje relativní četnost všech stupňů jasu. Jedná se o vektor, který má délku totožnou s počtem všech zobrazitelných stupňů šedi. Například pokud bychom měli hloubku obrazu 8 bitů, čili šedotónový obraz, délka vektoru by byla 28, čili 256 stupňů šedi. Největšího využití se histogram dočkal pro vizuální hodnocení technické kvality snímku. Jestliže je histogram úzký, je obraz málo kontrastní. Naopak pokud je histogram široký, je snímek příliš kontrastní. Pokud je maximum histogramu posunuto vlevo, je snímek příliš tmavý neboli podexponovaný. Naopak maximum posunuté doprava značí příliš jasný snímek neboli přeexponovaný [8].

3 Přehled funkcí pro zpracování 2D obrazu

Wolfram Mathematica obsahuje celou řadu funkcí, nejen z oblasti vědeckých, technických a matematických výpočtů. Přehled funkcí, které obsahuje Wolfram Language, je k dispozici online na Wolfram Language & System Documentation Center [22]. Každá funkce obsahuje zápis, její popis a jsou uvedeny i příklady aplikace dané funkce. Tato kapitola uvádí přehled a popis funkcí týkající se zpracování zejména 2D obrazu, při které bylo využito právě Wolfram Documentation Center. Jeho náhled uveden na obr. 1.

Obr. 1: *Documentation Center. Převzato a upraveno z [22].*

3.1 Vytvoření a import obrazu

Tato kapitola je zaměřena na tvorbu a import obrázků, aby jej bylo možné využít na další zpracování. Obraz lze buď importovat do Notebooku, nebo pomocí příslušných funkcí zachytit snímek v reálném čase a vložit jej přímo do Notebooku.

3.1.1 Import obrazu

Jelikož často budeme pracovat s obrázky, potřebujeme je vkládat do Notebooku. Pro vložení obrázku slouží funkce **Import**, ta vkládá obraz ve standardních formátech (tiff, png, jpeg, ...). Je nutné, aby byl obrázek ve stejné složce, jako je daný Notebook. Funkcí **ImageResize** upravíme obrázek na zvolenou velikost pro další práci. Dále můžeme vytvořit miniaturu za pomocí funkce **Thumbnail**.

In[1]:=
```
SetDirectory[NotebookDirectory[]];
```

In[2]:=
```
Obr1 = Import["obr.jpg"];
```

In[3]:=
```
Obr = ImageResize[Obr1, 100]
```

Out[3]=

In[4]:=
```
Thumbnail["Obr.jpg"]
```

Out[4]=

Přehled funkcí:

- Copy
- Import - Import[*source*], Import[*source,"format"*], Import[*source,elements*], Import[*source,elements*]
- Thumbnail - Thumbnail[*image*], Thumbnail[*file*], Thumbnail[*url*], Thumbnail[*spec,size*]

3.1.2 Vytvoření obrazu

Funkce **Image** generuje podle matice obrázek, **ColorSpace** ve funkci udává, jaký způsob míchání barev bude použit. Při použití aditivního míchání barev bude mít černá barva hodnoty 0, naopak pro získání bílé barvy jsou hodnoty na 1. Pro převedení obrazu do rastrové formy slouží funkce **Rasterize**. Funkce **ImageGraphic** vrátí obsah formou vektorové grafiky a pomocí argumentu lze zadat počet barev na výstupu.

In[5]:=
```
Image[{{{0.2, 0.7, 0.2}, {0.1, 0.8, 0.8}}, {{0, 0, 0}, {1, 1, 1}}}, ColorSpace → "RGB"]
```

Out[5]=

In[6]:=
```
Rasterize[Obr]
```

Out[6]=

In[7]:=
```
ImageGraphics[Obr, 8]
```

Out[7]=

Přehled funkcí:

- **Image, Image3D - Image**[*data*], **Image**[*graphics*], **Image**[*obj*,*options*]
- **Rasterize** - **Rasterize**[*expr*], **Rasterize**[*expr*,*elem*], **Rasterize**[*expr*,{*elem1*,*elem2*,...}]
- **ImageGraphics - ImageGraphics**[*image*], **ImageGraphics**[*image*,*n*], **ImageGraphics**[*image*,*colors*]

3.1.3 Parametrické generování obrazu

Funkci **ConstatImage** použijeme v situaci, kdy chceme vygenerovat obraz konstantní hodnoty nebo barvy. Obraz můžeme vygenerovat jak ve 2D, tak ve 3D a to za pomocí funkce **Image3D**. Funkci ConstantImage zapisujeme ve tvaru ConstatImage[val, size, “type”].

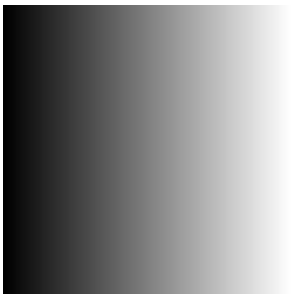
In[8]:=
```
ConstantImage[Blue, {50, 20}]
```

Out[8]=

Dále můžeme vygenerovat obrázek s měnícím se gradientem, a to buď s lineárním za pomocí funkce **LinearGradientImage** nebo radiálním **RadialGradientImage**. Funkce můžeme vyhodnotit s prázdným vstupem, tím pádem dostaneme vygenerován obrázek s černobílým gradientem.

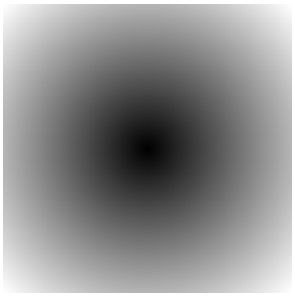
In[9]:=
```
LinearGradientImage[]
```

Out[9]=



In[10]:=
```
RadialGradientImage[]
```

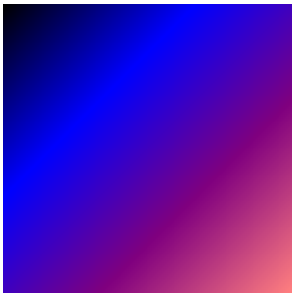
Out[10]=



Na funkci můžeme aplikovat barvy, či pozměnit směr gradientu, např.:

In[11]:=
```
LinearGradientImage[
 {{Right, Bottom}, {Left, Top}} -> {Pink, Purple, Blue, Black}]
```

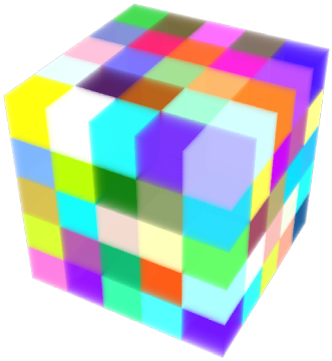
Out[11]=



Funkci **RandomImage** použijeme, když chceme vygenerovat obraz symbolického rozložení. Obraz můžeme vygenerovat i ve 3D a i v barvách.

In[12]:=
```
RandomImage[1.5, {5, 5, 5}, ColorSpace → "RGB"]
```

Out[12]=



Pro vytvoření čárového nebo QR kódu je určena funkce **BarcodeImage**.

In[13]:=
```
BarcodeImage["1234567", "QR"]
```

Out[13]=

Přehled funkcí:

- `ConstantImage` - `ConstantImage[val,size]`, `ConstantImage[val,size,"type"]`
- `LinearGradientImage`, `RadialGradientImage` - `LinearGradientImage[gcol]`, `LinearGradientImage[{pos1,pos2}→gcol]`, `LinearGradientImage[...,size]`
- `RandomImage` - `RandomImage[{min,max}]`, `RandomImage[dist]`, `RandomImage[...,size]`
- `BarcodeImage` - `BarcodeImage["string",format]`, `BarcodeImage["string",format,size]`

3.1.4 Pořízení obrazu

V této kapitole se podíváme na zachycování snímků v reálném čase. Pro zachycení obrazu použijeme funkci **CurrentImage**, která zachytí snímek. Více možností nabízí funkce **ImageCapture**, pomocí níž otevřeme uživatelské rozhraní a můžeme pořizovat jak snímky, tak i videa z kamery nebo jiného zařízení. Dále pomocí funkce **CurrentScreenImage** můžeme zachytit aktuální obraz obrazovky a **CurrentNotebookImage** zachytí obraz aktuálního Notebooku.

Přehled funkcí:

- `CurrentImage` - `CurrentImage[]`, `CurrentImage[n]`

- **ImageCapture - ImageCapture[]**
- **CurrentScreenImage - CurrentScreenImage[], CurrentScreenImage[*n*], CurrentScreenImage[*{{xmin,ymin},{xmax,ymax}}*]**
- **CurrentNotebookImage - CurrentNotebookImage[*nb*], CurrentNotebookImage[]**

3.2 Vlastnosti a zobrazení obrazu

Mathematica umožňuje znázornění a zpracování obrazu, podporuje obraz s libovolným počtem kanálů a barevnou hloubkou a celou řadou interních datových typů, které jsou specifikovány nebo automaticky zvoleny. Tato kapitola je zaměřena na funkce ohledně vlastností a možností obrazu, barev a hladinám barev. Jako poslední je popsán tzv. dynamický prohlížeč.

3.2.1 Vlastnosti obrazu

Funkce **ImageQ** testuje a vrací hodnotu True, jestliže je testovaný objekt obraz typu Image nebo Image3D a hodnotu False v ostatních případech.

```
In[14]:= ImageQ[Obr]
Out[14]= True
```

Pokud potřebujeme určit vlastnosti obrázku, použijeme funkci **ImageMeasurement**. Tato funkce vrací zadané vlastnosti obrázku, kterými můžou být např. "**Mean**" - průměrná barva; "**MeanIntensity**" - průměrná intenzita; "**DataType**" - typ dat; "**DataRange**" - rozsah dat, "**Min**" a "**Max**" - minimum a maximum.

```
In[15]:= ImageMeasurements[Obr, "Mean"]
Out[15]= {0.547629, 0.545456, 0.516161}
```

V některých případech můžeme potřebovat zjistit datový typ obrázku. K tomuto slouží funkce **ImageType**. Datové typy mohou být: Bit, Byte, Bit16, Real32, Real. Funkce **ImageData** vytvoří z obrázku datové pole.

```
In[16]:= ImageType[Obr]
Out[16]= Byte
```

Pokud chceme zjistit jakého typu barevného modelu obrázek je, využijeme funkci **ImageColorSpace** a funkci **ImageChannels** pro zjištění počtu kanálů. Rozměry zjistíme za pomocí funkce **ImageDimensions** a poměr výšky k šířce obrázku udává funkce **ImageAspectRatio**. **ImageValue** určí hodnotu pixelu v dané pozici, **ImageValuePositions** určí polohy daných pixelů.

```
In[17]:= ImageColorSpace[Obr]
Out[17]= RGB

In[18]:= ImageChannels[Obr]
Out[18]= 3

In[19]:= ImageDimensions[Obr]
Out[19]= {100, 100}

In[20]:= ImageAspectRatio[Obr]
Out[20]= 1
```

```
In[21]:= ImageValue[Obr, {10, 10}, Byte]
Out[21]= {32, 27, 16}

In[22]:= ImageValuePositions[Obr, Black]
Out[22]= {}
```

Přehled funkcí:

- **Image, Image3D** - viz kapitole 3.1.2
- **ImageQ - ImageQ**[*image*]
- **ImageMeasurements - ImageMeasurements**[*image,"prop"*],
ImageMeasurements[*image,"prop",format*],
ImageMeasurements[{*image*$_1$,*image*$_2$,...},...]
- **ImageType - ImageType**[*image*]
- **ImageData - ImageData**[*image*], **ImageData**[*image,"type"*]
- **ImageColorSpace - ImageColorSpace**[*image*]
- **ImageChannels - ImageColorSpace**[*image*]
- **ImageDimenstions - ImageDimensions**[*image*]
- **ImageAspectRatio - ImageAspectRatio**[*image*]
- **ImageValue**, **PixelValue - ImageValue**[*image,pos*], **ImageValue**[*image,pos,"type"*]
- **ImageValuePositions**, **PixelValuePositions** - **ImageValuePositions**[*image,val*],
ImageValuePositions[*image,val,d*]

3.2.2 Možnosti obrazu

Funkce **Options** zobrazí možnosti obrázku.

In[23]:=
```
Options[Obr]
```

Out[23]=
```
{ColorSpace → RGB, Interleaving → True, MetaInformation →
  ⟨|Exif → ⟨|ImageDescription → dav, Make → HUAWEI, Model → FIG-LX1,
    Orientation → ⟨|CameraTopOrientation → Top, Mirrored → False|⟩,
    XResolution → 72, YResolution → 72, ResolutionUnit → inch,
    Software → FIG-LX1 8.0.0.148(C432), DateTime → Fri 24 Aug 2018 13:34:11,
    ExposureTime → 0.000878 s, FNumber → f/2.2, ExposureProgram → Auto,
    ISOSpeedRatings → 50, ExifVersion → 2.10, DateTimeOriginal →
     Fri 24 Aug 2018 13:34:11, DateTimeDigitized → Fri 24 Aug 2018 13:34:11,
    ComponentsConfiguration → YCbCr, ShutterSpeedValue → 29.8973,
    ApertureValue → 2.2, BrightnessValue → 0.,
    ExposureBiasValue → 0. exposure values, MeteringMode → Multi-segment,
    LightSource → Other light source, FlashInfo → ⟨|FlashUsed → False,
      FlashFiringStatus → No strobe return detection function,
      FlashFunctionPresent → True, RedEyeCorrection → False|⟩,
    FocalLength → 3.5 mm, SubSecTime → 150966 ms,
    SubSecTimeOriginal → 150966 ms, SubSecTimeDigitized → 150966 ms,
    FlashpixVersion → 1.00, ColorSpace → RGBColor, PixelXDimension → 3050,
    PixelYDimension → 3050, SensingMethod → One-chip color area,
    FileSource → Digital still camera, SceneType → Directly photographed,
    CustomRendered → Custom process, ExposureMode → Auto, WhiteBalance → Auto,
    DigitalZoomRatio → 1., FocalLengthIn35mmFilm → 26 mm,
    SceneCaptureType → Standard, GainControl → None, Contrast → Normal,
    Saturation → Normal, Sharpness → Normal, SubjectDistanceRange → Unknown|⟩,
  IPTC → ⟨|Application2 → ⟨|Caption → dav, DateCreated → Day: Fri 24 Aug 2018,
     DigitizationDate → Day: Fri 24 Aug 2018,
     DigitizationTime → 13:34:11 GMT, RecordVersion → 2,
     TimeCreated → 13:34:11 GMT|⟩, Envelope → ⟨|CharacterSet → UTF8|⟩|⟩,
  XMP → ⟨|DublinCoreSchema → ⟨|Description → dav|⟩,
    PhotoshopSchema → ⟨|DateCreated → Fri 24 Aug 2018 13:34:11 GMT+2.|⟩,
    BasicSchema → ⟨|CreateDate → Fri 24 Aug 2018 13:34:11 GMT+2.,
     CreatorTool → FIG-LX1 8.0.0.148(C432),
     ModifyDate → Fri 24 Aug 2018 13:34:11 GMT+2.|⟩|⟩|⟩}
```

ColorSpace použijeme, když chceme změnit barevný model obrázku. Mezi podporovanými barevnými modely jsou Grayscale, RGB, CMYK, HSB, XYZ, LAB, LCH, LUV. Funkce **Magnification** umožňuje zvětšení nebo zmenšení obrázku.

In[24]:=
```
Image[Obr, ColorSpace -> "HSB", Magnification → 0.1]
```

Out[24]=

Funkce **Interleaving** je volba pro obraz a souvisejících funkcí, které určují, zda mají být data odpovídající různým kanálům v obraze prokládána. **MetaInformation** je možnost, která dává

meta-informace pro obraz Image, CloudObject a pro další objekty.

ImageSize je volba, která určuje celkovou velikost, na kterou se má zobrazit daný objekt, obraz. Volba **ImageResolution** je určen pro Export, Rasterize a pro související funkce, které určují, v jakém rozlišení bitmapových obrázků by měly být vykresleny.

```
In[25]:= Table[Rasterize[Style[x^2 + y^2, 40], ImageResolution → r], {r, {10, 20, 30}}]
```

Out[25]= $\{\sqrt{x^2+y^2}, x^2+y^2, x^2+y^2\}$

```
In[26]:= {√(x^2 + y^2), x^2 + y^2, x^2 + y^2}
```

Out[26]= $\{\sqrt{x^2+y^2}, x^2+y^2, x^2+y^2\}$

Přehled funkcí:

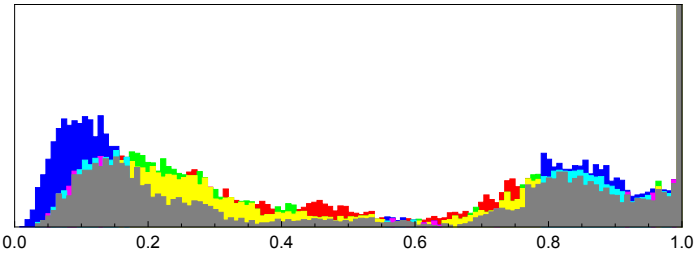
- `Options - Options[symbol], Options[expr], Options[stream], Options["sname"], Options[object], Options[obj,name], Options[obj,{name1,name2,...}]`
- `ColorSpace`
- `Interleaving`
- `MetaInformation`
- `ImageSize`
- `Magnification`
- `ImageResolution`

3.2.3 Barvy a hladiny barev

Histogram jednotlivých úrovní barev, který obrázek obsahuje vytvoříme za pomocí funkce **ImageHistogram**. **DominantColors** vypíše seznam dominantních barev, které se v obrázku vyskytují.

```
In[27]:= ImageHistogram[Obr]
```

Out[27]=



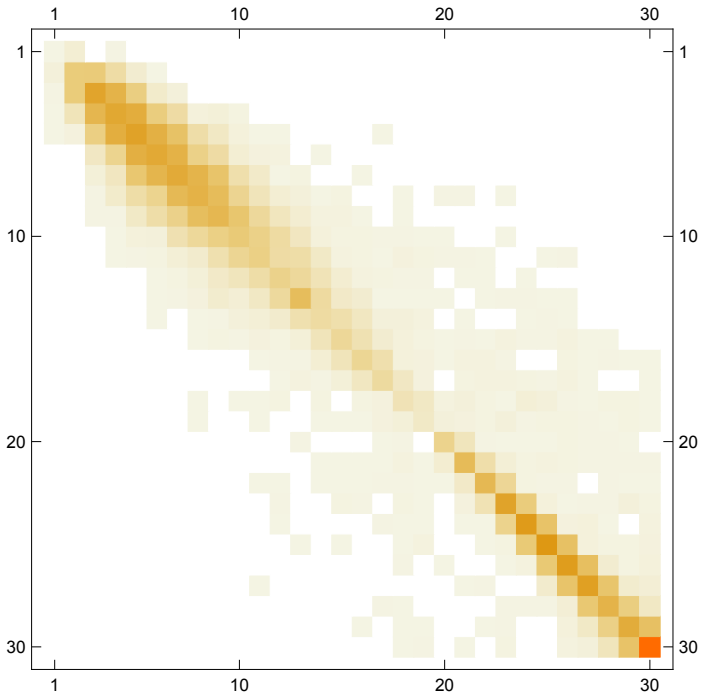

```
In[28]:= DominantColors[Obr]
```

Out[28]= {□, ■, ■, ■, ■}

K zobrazení matice výskytu intenzity pixelu se využívá funkce **ImageCooccurrence**. **BinaryImageQ** vrací hodnotu True, pokud má obraz binární podobu, v jiných případech vrací hodnotu False.

In[29]:=
```
MatrixPlot[ImageCooccurrence[Obr, 30]]
```

Out[29]=



In[30]:=
```
BinaryImageQ[Obr]
```

Out[30]=
```
False
```

FindTreshold najde globální prahovou hodnotu, která rozděluje úroveň intenzity. **Binarize** vytvoří binární obraz tak, že nahradí všechny hodnoty nad globálním prahem.

In[31]:=
```
FindThreshold[Obr]
```

Out[31]=
```
0.54902
```

In[32]:=
```
Binarize[Obr]
```

Out[32]=

ImageLevels poskytuje seznam hodnot pixelů a počet pro každý kanál v obraze. Funkce **ColorConvert** převede obraz do daného barevného modelu.

In[33]:=
```
ImageLevels[Obr, 5]
```

Out[33]=
```
{{{0., 2274}, {0.2, 2089}, {0.4, 828}, {0.6, 1388}, {0.8, 3421}},
 {{0., 2426}, {0.2, 2164}, {0.4, 582}, {0.6, 1137}, {0.8, 3691}},
 {{0., 3859}, {0.2, 930}, {0.4, 386}, {0.6, 843}, {0.8, 3982}}}
```

In[34]:=
```
ColorConvert[Obr, "Grayscale"]
```

Out[34]=

AlphaChannel vrací alfa kanál obrazu. Funkce **ColorSeparate** rozloží obraz na jednotlivé jednokánalové obrazy, kde každý obraz odpovídá jednomu barevnému kanálu v obraze. Opak funkce je **ColorCombine**. Ta slouží k vytvoření vícekanálového obrazu kombinací sekvencí kanálů v obraze. Lze použít též pro kombinaci obrazů, které představují barevné komponenty určené barevným prostorem.

In[35]:=
```
ColorSeparate[Obr]
```

Out[35]=

In[36]:=
```
ColorCombine[{ , , }, "RGB"]
```

Out[36]=

Přehled funkcí:

• **ImageHistogram - ImageHistogram**[*image*], **ImageHistogram**[*image,bspec*], **ImageHistogram**[*image,bspec,range*]

• **FindThreshold - FindThreshold**[*image*]

• **DominantColors - DominantColors**[*image*], **DominantColors**[*image,n*], **DominantColors**[*image,n,prop*], **DominantColors**[*image,n,prop,format*], **DominantColors**[{$image_1$,$image_2$,...},...]

• **ImageCooccurrence - ImageCooccurrence**[*image,n*], **ImageCooccurrence**[*image,n,ker*]

• **BinaryImageQ - BinaryImageQ**[*image*]

• **Binarize - Binarize**[*image*], **Binarize**[*image,t*], **Binarize**[*image*,{t_1,t_2}], **Binarize**[*image,f*]

• **ImageLevels - ImageLevels**[*image*], **ImageLevels**[*image,bspec*], **ImageLevels**[*image,bspec,range*]

• **ColorConvert - ColorConvert**[*expr,colspace*]

• **AlphaChannel - AlphaChannel**[***image***]

• **ColorSeparate - ColorSeparate**[*image*], **ColorSeparate**[*image*], **ColorSeparate**[*image,channel*]

• **ColorCombine - ColorCombine**[{$image_1$,$image_2$,...}], **ColorCombine**[{$image_1$,$image_2$,...},*colorspace*]

3.2.4 Dynamický prohlížeč

DynamicImage zobrazí dynamický prohlížeč, díky němu lze prohlížet obrázek, soubor nebo URL stránku. **ZoomCenter** je volba pro **DynamicImage**, která určuje polohu okna při přiblížení obrazu. Další volba pro **DynamicImage** je **ZoomFactor**, která určuje zvětšení zoomu.

In[37]:=
```
DynamicImage[Obr, ZoomCenter → {Center, Top}, ZoomFactor → 0.2]
```

Out[37]=

Přehled funkcí:

- **DynamicImage** - **DynamicImage**[*image*], **DynamicImage**[*file*], **DynamicImage**[*url*]

3.3 Základní manipulace s obrazem

Někdy při zpracování můžeme potřebovat pouze určitý výřez obrázku. Právě tato kapitola je zaměřena na strukturální operace, díky kterým lze například oříznout rozměry obrázku. Jako další budou následovat geometrické operace, orientované zejména na rotaci obrazu a dále kompozice obrazu. V dalších částech je popsáno základní zpracování obrazu, zabývající se především nastavením úrovně jasu, kontrastu a gamy. V poslední části jsou popsány funkce na práci s alfa kanálem a funkce pro operace s pixely.

3.3.1 Strukturální operace

Jestliže budeme potřebovat obrázek oříznout, využijeme funkci **ImageCrop**. Naopak funkce **ImageTrim** nám ořízne obrázek na nejmenší možnou plochu tak, že obsahuje všechny zadané body.

In[38]:=
```
ImageCrop[Obr, {50, 50}]
```

Out[38]=

In[39]:=
```
ImageTrim[Obr, {{10, 10}, {50, 50}}]
```

Out[39]=

Jestliže chceme přidat okraje či oříznout obrázek, využijeme funkci **ImagePad**. Můžeme také zadat, že na každé straně obrázku bude jiný počet pixelů k ohraničení. Jestliže zadáme kladné hodnoty, přidáme tím ohraničení. Naopak při zadání záporné hodnoty, dojde k oříznutí

okraje. Pomocí funkce **ImageTake** vybereme část obrázku, která obsahuje následující body. Můžeme též zvolit, mezi kterým řádkem a sloupcem má být část vybrána.

In[40]:=
```
ImagePad[Obr, {{ 10, 10}, {-20, -20}}]
```

Out[40]=

In[41]:=
```
ImageTake[Obr, {20, 80}, {50, 100}]
```

Out[41]=

Pro zobrazení šířky pixelů jednotlivých okrajů využijeme funkci **BorderDimensions**, která vyhodnotí okraje ve tvaru {{*left*, *right*}, {*bottom*, *top*}}.

In[42]:=
```
okraje = ImagePad[Obr, {{ 50, 40}, {30, 20}}]
```

Out[42]=

In[43]:=
```
BorderDimensions[okraje]
```

Out[43]=
```
{{50, 40}, {30, 20}}
```

Přehled funkcí:

- **ImageCrop** - **ImageCrop**[*image*], **ImageCrop**[*image*,*size*], **ImageCrop**[*image*,*size*,*spec*]
- **ImageTrim** - **ImageTrim**[*image*,*roi*], **ImageTrim**[*image*,*roi*,*r*], **ImageTrim**[*image*,{*roi*$_1$,*roi*$_2$,...},...]
- **ImagePad** - **ImagePad**[*image*,*m*], **ImagePad**[*image*,*m*,*padding*], **ImagePad**[*image*,{{*left*,*right*},{*bottom*,*top*}},...], **ImagePad**[*image*,{{*left*,*right*},{*front*,*back*},{*bottom*,*top*}},...]
- **ImageTake** - **ImageTake**[*image*,*n*], **ImageTake**[*image*,-*n*], **ImageTake**[*image*,-*n*], **ImageTake**[*image*,{*row*$_1$,*row*$_2$},{*col*$_1$,*col*$_2$}], **ImageTake**[*image3d*,{*slice*$_1$,*slice*$_2$},{*row*$_1$,*row*$_2$},{*col*$_1$,*col*$_2$}]
- **BorderDimensions** - **BorderDimensions**[*image*], **BorderDimensions**[*image*,*t*]

3.3.2 Geometrické operace

Na zmenšení obrázku je vhodná funkce **ImageResize**, která změní velikost převzorkováním jeho rastrového vzoru, poskytuje zmenšení obrázku na zadanou šířku pixelů.

In[44]:=
```
ImageResize[Obr, 70]
```

Out[44]=

K otočení obrázku slouží funkce **ImageRotate**. Tato funkce buď otáčí obrázek o úhel 90°, nebo ho lze otočit o požadovaný úhel. Funkci je možné aplikovat i ve 3D.

In[45]:=
```
ImageRotate[Obr, 30 Degree]
```

Out[45]=

Funkce **ImageReflect** zrcadlově obrazí obraz. Standartně obrací shora dolů. Pokud je ve tvaru ImageReflect[*mage*, $side_1 \to side_2$] provádí zrcadlení strany 1 na stranu 2.

In[46]:=
```
ImageReflect[Obr, Top → Left]
```

Out[46]=

Přehled funkcí:

- **ImageResize** - **ImageResize**[*image*,*width*], **ImageResize**[*image*,{*s*}], **ImageResize**[*image*,{*width*,*height*}], **ImageResize**[*image*,{*width*,*depth*,*height*}]
- **Thumbnail** - viz kapitola 3.1.1
- **ImageRotate** - **ImageRotate**[*image*], **ImageRotate**[*image*,θ], **ImageRotate**[*image*,{θ,*w*}], **ImageRotate**[*image*,...,*size*]
- **ImageReflect** - **ImageReflect**[*image*], **ImageReflect**[*image*,*side*], **ImageReflect**[*image*,$side_1 \to side_2$]

3.3.3 Kompozice obrazu

Ke skládání obrázku přes sebe překrytím je funkce **ImageCompose**. Můžeme též definovat, kde se bude překrytí na obrázku nacházet. **ImageAssemble** dokáže z řady obrázků složit obraz. **ImageCollage** vytváří koláž z obrázků, můžeme též nastavit poměry mezi jednotlivými obrázky v koláži.

In[47]:=
```
ImageCollage[{1 → Obr, 4 → Obr, 1 → Obr}]
```

Out[47]=

ImageAdd přidá zvolenou hodnotu jasu ke každé hodnotě kanálu v obraze či lze pomocí této funkce složit obraz, ve kterém každý pixel bude součtem odpovídajících pixelů v jednotlivých obrázcích. Opakem této funkce je **ImageSubtract**, který danou hodnotu od obrazu odečítá či poskytuje obraz, ve kterém je každý pixel získán odečtením hodnot odpovídajících pixelů v jednotlivých obrazech. Funkce **ImageMultiply** vynásobí každou hodnotu kanálu zvoleným faktorem. Dále poskytuje obraz, ve kterém je každý pixel součinem odpovídajících pixelů v jednotlivých obrazech. Výsledkem funkce **ImageDifference** je obraz, ve kterém je každý pixel

absolutním rozdílem pixelů v jednotlivých obrazech.

In[48]:=
```
ImageAdd[Obr, 0.25]
```

Out[48]=

In[49]:=
```
ImageSubtract[Obr, 0.25]
```

Out[49]=

In[50]:=
```
ImageMultiply[Obr, 0.25]
```

Out[50]=

Přehled funkcí:

• **ImageCompose - ImageCompose**[*image*,*overlay*], **ImageCompose**[*image*,{*overlay*,α}], **ImageCompose**[*image*,*overlay*,*pos*], **ImageCompose**[*image*,*overlay*,*pos*,*opos*], **ImageCompose**[*image*,*overlay*,*pos*,*opos*,{f_i,f_o,*mode*}]

• **ImageAssemble - ImageAssemble**[{{$image_{11}$,$image_{12}$,...},{$image_{21}$,...},...}]

• **ImageCollage - ImageCollage**[{$image_1$,$image_2$,...}], **ImageCollage**[{w_1→$image_1$,w_2→$image_2$,...}], **ImageCollage**[⟨|$image_1$→w_1,$image_2$→w_2,...|⟩], **ImageCollage**[{w_1,w_2,...}→{$image_1$,$image_2$,...}], **ImageCollage**[{{$image_1$,w_1},{$image_2$,w_2},...}], **ImageCollage**[...,*fitting*], **ImageCollage**[...,*fitting*,*size*]

• **ImageAdd - ImageAdd**[*image*,*x*], **ImageAdd**[$image_1$,$image_2$], **ImageAdd**[*image*,$expr_1$,$expr_2$,...]

• **ImageSubtract - ImageSubtract**[*image*,*x*], **ImageSubtract**[$image_1$,$image_2$], **ImageSubtract**[**image**,$expr_1$,$expr_2$,...]

• **ImageMultiply** - **ImageMultiply**[*image*,*x*], **ImageMultiply**[$image_1$,$image_2$], **ImageMultiply**[*image*,$expr_1$,$expr_2$,...]

• **ImageDifference - ImageDifference**[$image_1$,$image_2$]

3.3.4 Základní zpracování obrazu

K nastavení úrovně jasu, kontrastu a gamy využijeme funkci **ImageAdjust**. Pokud nezadáme parametr, provádí se automatické nastavení. Nastavení úrovní můžeme provést i manuálně.

In[51]:=
```
ImageAdjust[Obr]
```

Out[51]=

Můžeme i aplikovat speciální efekty na obrázek využitím funkce **ImageEffect**.

In[52]:=
```
ImageEffect[Obr, "Comics"]
```

Out[52]=

Dále můžeme někdy potřebovat zaostřit nebo naopak rozostřit obrázek. K zaostření využijeme funkci **Sharpen**, naopak k rozostření funkci **Blur**. Funkce **Lighter** nám obrázek zesvětlí, ovšem při použití funkce **Darker** dojde ke ztmavení obrázku.

In[53]:=
```
Sharpen[Obr]
```

Out[53]=

In[54]:=
```
Blur[Obr]
```

Out[54]=

In[55]:=
```
Lighter[Obr]
```

Out[55]=

In[56]:=
```
Darker[Obr]
```

Out[56]=

Funkce **ImagePartition** rozděluje obraz do pole.

In[57]:=
```
ImagePartition[Obr, 20]
```

Out[57]=

Přehled funkcí:

• **ImageAdjust - ImageAdjust[***image***], ImageAdjust[***image,corr***], ImageAdjust[***image,corr,*{in_{min},in_{max}}**], ImageAdjust[***image,corr,*{in_{min},in_{max}}*,*{out_{min},out_{max}}**]**

- **ImageEffect - ImageEffect**[*image,"effect"*], **ImageEffect**[*image,{"effect",params}*]
- **Sharpen - Sharpen**[*image*], **Sharpen**[*image,r*]
- **Blur - Blur**[*image*], **Blur**[*image,r*]
- **Lighter - Lighter**[*color*], **Lighter**[*color,f*], **Lighter**[*image,...*]
- **Darker - Darker**[*color*], **Darker**[*color,f*], **Darker**[*image,...*]
- **ImagePartition - ImagePartition**[*image,s*], **ImagePartition**[*image,{w,h}*], **ImagePartition**[*image,{w,h},{dw,dh}*]
- **ImageAssemble** - viz kapitola 3.3.3

3.3.5 Operace s alfa kanálem

Funkce **AlphaChannel** a **ImageCompose** jsou popsány v předešlých kapitolách. **SetAlphaChannel** přidá do obrázku zcela neprůhledný alfa kanál. Lze též nastavit krytí pixelů. Opakem je funkce **RemoveAlphaChannel**, která odebere všechny informace o průhlednosti z obrázku či odstraní neprůhlednost smícháním všech pixelů s barvou pozadí.

In[58]:=
```
SetAlphaChannel[Obr, Graphics[{Gray, Disk[]}]]
```

Out[58]=

Přehled funkcí:

- **AlphaChannel** - viz kapitola 3.2.3
- **SetAlphaChannel - SetAlphaChannel**[*image*], **SetAlphaChannel**[*image,a*], **SetAlphaChannel**[*image,aimage*]
- **RemoveAlphaChannel - RemoveAlphaChannel**[*image*], **RemoveAlphaChannel**[*image,c*]
- **ImageCompose** - viz kapitola 3.3.3

3.3.6 Operace s pixely

Převážnou část funkcí popisující operace s pixely byla popsána v kapitole 3.2.1. Dalšími funkcemi jsou **ReplaceImageValue** a **ReplacePixelValue**, které změní hodnoty pixelů na dané pozici na obrázku na zadanou hodnotu.

In[59]:=
```
ReplaceImageValue[Obr, {100, 100} -> Black]
```

Out[59]=

ImageApply, ImageApplyIndexed aplikují libovolnou funkci na seznam hodnot kanálu pro každý pixel v obraze nebo lze aplikovat na danou sekvenci pixelů v obraze. Funkce **ImageScan** skenuje každý pixel obrazu s použitím zvolené funkce.

In[60]:= `ImageApply[Max, Obr]`

Out[60]=

In[61]:= `ImageScan[Print, ]`

```
0.414295
0.949619
0.668444
0.728566
0.183734
0.00973788
0.0186841
0.252154
0.758881
```

Přehled funkcí:

- **ImageValue, PixelValue** – viz kapitola 3.2.1
- **ReplaceImageValue, ReplacePixelValue – ImageValuePositions**[*image*,*val*], **ImageValuePositions**[*image*,*val*,*d*]
- **ImageValuePositions, PixelValuePositions** – viz kapitola 3.2.1
- **ImageApply, ImageApplyIndexed – ImageApply**[*f*,*image*], **ImageApply**[*f*,{*image*$_1$,*image*$_2$,...}]
- **ImageScan – ImageScan**[*f*,*image*]

3.4 Barevné zpracování obrazu

Wolfram Language poskytuje funkce a algoritmy pro manipulaci s barevným obrazem. V této kapitole jsou popsány funkce pro základní operace s barevným obrazem, dále funkce pro nastavení barev, zpracování úrovní a tvorba histogramu. Následně jsou popsány barevné modely a operace s kanály a pixely. Jako poslední jsou popsány funkce pro pseudobarevný obraz.

3.4.1 Základní operace

Pro negaci barev slouží funkce **ColorNegate**. Můžeme negovat jak obrázek, tak i barvu. Jestliže potřebujeme obrázek zabarvit či přebarvit, využijeme funkci **Blend**. Funkce **ImageDemosaic** rekonstruuje obraz pomocí specifického pole barevných filtrů, zarovnává levý horní pixel vzoru s obrazem.

In[62]:= `ColorNegate[Obr]`

Out[62]=

In[63]:=
```
Blend[{Obr, Blue}]
```

Out[63]=

Přehled funkcí:

- **ColorNegate** – **ColorNegate**[*image*], **ColorNegate**[*color*]
- **Blend** – **Blend**[{col_1,col_2},x], **Blend**[{{x_1,col_1},{x_2,col_2},…},x], **Blend**[{col_1,col_2,…},{u_1,u_2,…}], **Blend**[{$image_1$,$image_2$,…},…]
- **ImageDemosaic** – **ImageDemosaic**[*image*,{"cfa",{*row*,*col*}}]

3.4.2 Nastavení barev

Jestliže máme obrázek s nerovnoměrným jasem či kontrastem, využijeme funkce **BrightnessEqualize**. Ta rovnoměrně upraví úroveň jasu, kontrastu, gamy atd.

In[64]:=
```
BrightnessEqualize[Obr]
```

Out[64]=

ColorBalance upraví barvy na obrázku tak, aby bylo dosaženo barevné rovnováhy při neutrálním osvětlení nebo upraví barvy tak, aby zadaná referenční barva byla změněna na bílou. Funkce **ColorToneMapping** aplikuje mapování tónů na hodnoty barev v obraze. Změny jasu jsou viditelné i v malých intervalech dynamického rozsahu.

In[65]:=
```
ColorBalance[Obr]
```

Out[65]=

In[66]:=
```
ColorToneMapping[Obr]
```

Out[66]=

Přehled funkcí:

- **ImageAdjust** – viz kapitola 3.3.4
- **BrightnessEqualize** – **BrightnessEqualize**[*image*], **BrightnessEqualize**[*image*,*flatfield*], **BrightnessEqualize**[*image*,*flatfield*,*darkfield*]
- **Lighter** – viz kapitola 3.3.4
- **Darker** – viz kapitola 3.3.4
- **ColorBalance** – **ColorBalance**[*image*], **ColorBalance**[*image*,*ref*],

ColorBalance[*image*,*ref*→*target*]

• **ColorToneMapping** - **ColorToneMapping**[*image*], **ColorToneMapping**[*image*,*c*], **ColorToneMapping**[*image*,*range*], **ColorToneMapping**[*image*,{*range*,*c*}], **ColorToneMapping**[*image*,{{$range_1$,c_1},{$range_2$,c_2},…}], **ColorToneMapping**[*image*,*spec*,*s*]

3.4.3 Zpracování úrovní, histogram

Pomocí funkce **ColorQuantize** lze aproximovat obraz kvantováním do odlišných barev. Lze použít i více možných barev. **Treshold** nahradí prahové hodnoty hodnotami blízkých nule nebo pomocí prahové specifikace.

In[67]:=
```
ColorQuantize[Obr, {Black, LightBlue}]
```

Out[67]=

In[68]:=
```
Threshold[Obr, {"Hard", "Cluster"}]
```

Out[68]=

Vyrovnání úrovní histogramu můžeme provést pomocí funkcí **HistogramTransform** nebo **HistogramTransformInterpolation**.

In[69]:=
```
HistogramTransform[Obr]
```

Out[69]=

Přehled funkcí:

• **ImageHistogram** - viz kapitola 3.2.3

• **ImageLevels** - viz kapitola 3.2.3

• **DominantColors** - viz kapitola 3.2.3

• **Binarize** - viz kapitola 3.2.3

• **ColorQuantize** - **ColorQuantize**[*image*], **ColorQuantize**[*image*,*n*], **ColorQuantize**[*image*,{col_1,…,col_2}]

• **Threshold** - **Threshold**[*data*], **Threshold**[*data*,*tspec*], **Threshold**[*image*,…], **Threshold**[*sound*,…]

• **FindTreshold** - viz 3. 2. 3

• **HistogramTransform, HistogramTransformInterpolation** - **HistogramTransform**[*image*], **HistogramTransform**[*image*,*dist*], **HistogramTransform**[*image*,*ref*], **HistogramTransform**[{x_1,x_2,…},…]

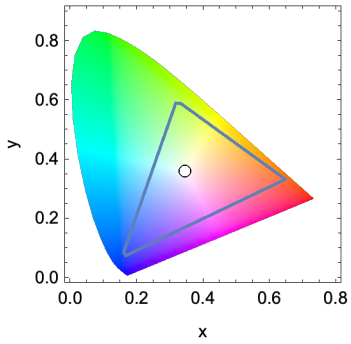
3.4.4 Barevné modely a prostory

Funkce **ColorSpace** aplikujeme na možnosti barevných prostorů. Barevné modely používané v Mathematice jsou následující: Grayscale, RGB, CMYK, HSB, XYZ, LAB, LCH, LUV. Zobrazit

jednotlivé barevné modely lze pomocí funkce **ChromaticityPlot**. Pokud si chceme prohlédnout barevný model v prostoru, slouží k tomu funkce **ChromaticityPlot3D**.

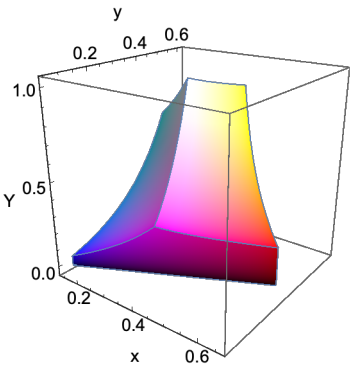
In[70]:=
```
ChromaticityPlot["RGB", ImageSize → Small]
```

Out[70]=



In[71]:=
```
ChromaticityPlot3D["RGB", ImageSize → Small]
```

Out[71]=



Přehled funkcí:

- `ColorConvert` – viz kapitola 3. 2. 3
- `ColorSpace`
- `ImageColorSpace` – viz kapitola 3.2.1
- `ColorProfileData` – viz kapitola 3.2.2
- `ChromaticityPlot, ChromaticityPlot3D` – `ChromaticityPlot[`*colspace*`]`, `ChromaticityPlot[`*color*`]`, `ChromaticityPlot[{`$col_1, col_2, \ldots$`}]`, `ChromaticityPlot[`*image*`]`, `ChromaticityPlot[`*…*`,`*refcolspace*`]`

3.4.5 Operace s kanály

Pro práci s kanály se využívají funkce popsané v předešlých kapitolách.

Přehled funkcí:

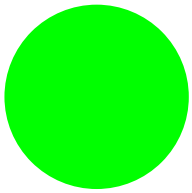
- `ColorSeparate` – viz kapitola 3.2.3
- `ColorCombine` – viz kapitola 3.2.3
- `ImageChannels` – viz kapitola 3.2.1
- `AlphaChannel` – viz kapitola 3.3.5
- `SetAlphaChannel` – viz kapitola 3.3.5
- `RemoveAlphaChannel` – viz kapitola 3.3.5

3.4.6 Operace s pixely

Pro nahrazování barev jinou barvou využijeme funkci **ColorReplace**, kde zadáme, kterou barvu chceme jakou barvou nahradit.

In[72]:=
```
kolečko = Graphics[{Green, Disk[]}, ImageSize → Tiny]
```

Out[72]=



In[73]:=
```
ColorReplace[kolečko, Green → Blue]
```

Out[73]=

Přehled funkcí:

- **ImageValuePositions, PixelValuePositions** – viz kapitola 3.2.1
- **ImageValue, PixelValue** – viz kapitola 3.2.1
- **ReplaceImageValue, ReplacePixelValue** – viz kapitola 3.3.6
- **ColorReplace** – **ColorReplace**[*image*,*color*], **ColorReplace**[*image*,*color*→*replacement*], **ColorReplace**[*image*,*color*→*replacement*,*d*], **ColorReplace**[*image*,{*color*$_1$→*replacement*$_1$,...},{*d*$_1$,...}]
- **ImageApply** – viz kapitola 3.3.6
- **ImageScan** – viz kapitola 3.3.6

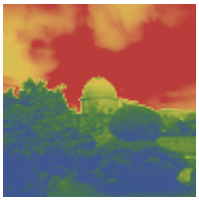
3.4.7 Pseudobarevný obraz

Jestliže máme obraz ve stupních šedé, můžeme využít funkci **Colorize**, která nahradí v obraze hodnoty intenzity hodnotami pseudobarev. **ReliefImage** vytvoří reliéfový obraz z řady hodnot. Ke zvýraznění vybraných oblastí v obrázku slouží funkce **HighlightImage**. Lze zvýraznit více vybraných oblastí.

In[74]:=
```
Colorize[
```

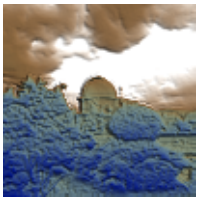
```
]
```

Out[74]=



In[75]:=
```
ReliefImage[Obr]
```

Out[75]=



In[76]:=
```
HighlightImage[Obr, ImageCorners[Obr, 1, .001, 5]]
```

Out[76]=

Přehled funkcí:

- **Colorize** – **Colorize**[*m*], **Colorize**[*image*]
- **ReliefImage** – **ReliefImage**[*array*]
- **HighlightImage** – **HighlightImage**[*image*,*roi*], **HighlightImage**[*image*,{*roi*$_1$,*roi*$_2$,...}], **HighlightImage**[*image*,{...,*w*[*roi*$_i$],...}], **HighlightImage**[*image*,*fg*,*bgstyle*]

3.4.8 Barvy

Při generování obrazců můžeme použít i barvu, která není v základních barvách funkcí. Tuto barvu vygenerujeme pomocí funkce **RGBColor**, kde zadáme hodnotu červené, zelené a modré. Podobně generuje i funkce **LABColor**, která pracuje s barevnými komponenty, lze ji též doplnit o složku neprůhlednosti. Pro zobrazení ve stupních šedi slouží funkce **GrayLevel**.

In[77]:=
```
Graphics[{RGBColor[0.5, 0.3, 0.9], Disk[]}, ImageSize → Tiny]
```

Out[77]=

Na generování náhodných barev slouží funkce **RandomColor**. **ColorQ** vrací hodnotu True, jestliže je barva z vybraných barevných prostorů, jinak vrací hodnotu False.

In[78]:=
```
RandomColor[10]
```

Out[78]= {, , , , , , , , , }

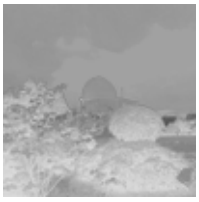
In[79]:=
```
ColorQ[Pink]
```

Out[79]=
```
True
```

ColorDistance udává přibližnou vzdálenost mezi jednotlivými barvami, prvky či poskytuje obrázek, jehož hodnoty pixelů jsou barevné vzdálenosti mezi pixely v obraze a zvolenou barvou. Nastavování barev umožňuje funkce **ColorSetter**, kdy zobrazí vzor zadané barvy a po kliknutí na vzor se otevře dialogové okno pro výběr barvy.

In[80]:=
```
ColorDistance[Obr, Pink]
```

Out[80]=



In[81]:=
```
ColorSetter[Yellow]
```

Out[81]=

Přehled funkcí:

- **RGBColor** - **RGBColor**[*red,green,blue*], **RGBColor**[*r,g,b,a*], **RGBColor**[*"string"*]
- **LABColor** - **LABColor**[*l,a,b*], **LABColor**[*l,a,b,α*]
- **GrayLevel** - **GrayLevel**[*level*], **GrayLevel**[*g,a*]
- **RandomColor** - **RandomColor**[], **RandomColor**[*n*], **RandomColor**[*model*], **RandomColor**[*model, n*], **RandomColor**[*model, {n₁, n₂,...}*]
- **ColorQ** - **ColorQ**[*color*]
- **ColorDistance** - **ColorDistance**[*c₁,c₂*], **ColorDistance**[*list,c*], **ColorDistance**[*list₁,list₂*], **ColorDistance**[*image,c*], **ColorDistance**[*image₁,image₂*]
- **ColorSetter** - **ColorSetter**[*color*], **ColorSetter**[**Dynamic**[*color*]], **ColorSetter**[]

3.5 Geometrické operace

Wolfram Language obsahuje základní, ale i sofistikované funkce pro geometrickou manipulaci s obrazem a transformace obrazu. V této kapitole nalezneme i funkce související se záznamem obrazu.

3.5.1 Základní geometrické operace

Základní geometrické operace s obrázky jsme již popsali v předešlé kapitole 3.3 Základní manipulace s obrazem.

Přehled funkcí:

- **ImageCrop** - viz kapitola 3.3.1
- **ImageTake** - viz kapitola 3.3.1
- **ImageTrim** - viz kapitola 3.3.1
- **ImagePad** - viz kapitola 3.3.1
- **ImageResize** - viz kapitola 3.3.2
- **ImageRotate** - viz kapitola 3.3.2
- **ImageReflect** - viz kapitola 3.3.2
- **ImagePartition** - viz kapitola 3.3.4
- **ImageAsseble** - viz kapitola 3.3.3

3.5.2 Geometrické transformace

Funkce **ImageTransformation** poskytuje obrázek, na který aplikuje zvolenou transformaci. Lze též upravit obrázek na požadovanou velikost. Obdobnou funkcí je **ImageForwardTransformation**, která poskytuje obrázek, ve které každý pixel v poloze `f[{x,y}]` odpovídá na obrázku pozici `{x,y}`. Poslední funkcí je **ImagePerspectiveTransformation**, která aplikuje lineární frakční transformaci specifickou maticí na každý pixel v obraze.

In[82]:=
```
ImageTransformation[Obr, Sqrt]
```

Out[82]=

In[83]:= `ImagePerspectiveTransformation[Obr, {{1, 0, 0}, {0, 1, 0}, {1, 0, 1}}]`

Out[83]=

Přehled funkcí:

- **ImageTransformation - ImageTransformation[*image,f*], ImageTransformation[*image,f,size*]**
- **ImageForwardTransformation - ImageForwardTransformation[*image,f*], ImageForwardTransformation[*image,f,size*]**
- **ImagePerspectiveTransformation - ImagePerspectiveTransformation[*image,m*], ImagePerspectiveTransformation[*image,tf*], ImagePerspectiveTransformation[*image,…,size*]**

3.5.3 Obrazový záznam

Funkce **ImageAlign** vrátí verzi obrázku, která je zarovnána s referenčním obrázkem. Můžeme též aplikovat na 3D obrázek. **ImageDisplacements** poskytuje vodorovné a svislé posunutí mezi po sobě jdoucími obrázky. Pro zjištění odpovídajících bodů ve dvou obrazech a vrácení jejich souřadnic pixelů využijeme funkci **ImageCorrespondingPoints**.

In[84]:= `ImageCorrespondingPoints[`  `,`  `]`

Out[84]=

```
{{{106.007, 139.964}, {96.72, 163.863}, {180.242, 158.152}, {204.13, 135.753},
  {182.415, 142.099}, {132.116, 167.043}, {116.469, 125.084}, {140.98, 104.847},
  {159.186, 133.353}, {188.472, 148.567}, {132.364, 185.883},
  {112.089, 96.453}, {156.348, 142.57}, {139.326, 125.104}, {156.854, 189.821},
  {110.159, 113.68}, {200.378, 121.04}, {156.421, 223.665}, {130.969, 133.363},
  {154.162, 147.439}, {246.613, 127.847}, {268.564, 258.376},
  {276.898, 265.608}, {214.59, 173.708}, {141.432, 227.552}, {125.841, 251.567},
  {116.394, 284.08}, {216.675, 180.268}, {212.491, 211.448}, {226.533, 220.}},
 {{37.6639, 66.2099}, {30.8566, 90.1901}, {108.665, 84.4464},
  {140.387, 61.0905}, {110.82, 67.7206}, {59.5454, 94.2705}, {45.8045, 51.1701},
  {66.3577, 37.3028}, {84.7667, 59.4187}, {118.214, 73.1216}, {54.0476, 109.89},
  {41.8118, 33.2554}, {82.5109, 68.1926}, {66.8978, 52.0239}, {84.7133, 117.642},
  {40.6218, 43.5324}, {134.222, 49.1353}, {81.289, 167.242}, {58.3286, 59.8075},
  {78.813, 72.4134}, {203.865, 53.5342}, {240.411, 222.538}, {256.203, 235.905},
  {154.282, 100.906}, {66.7183, 172.204}, {51.9292, 210.163}, {148.243, 283.636},
  {157.136, 107.901}, {144.513, 157.518}, {156.445, 159.152}}}
```

FindGeometricTransform nalezne geometrickou transformaci, která zarovná určené pozice a vrátí chybu zarovnání spolu s transformační funkcí. Pro výpočet vzdálenosti mezi dvěma obrázky se využívá funkce **ImageDistance**.

In[85]:= `ImageDistance[`  `,`  `]`

Out[85]= 194.224

Přehled funkcí:

- **ImageAlign - ImageAlign[*ref*,*image*], ImageAlign[*ref*,{*image*$_1$,…,*image*$_n$}], ImageAlign[{*image*$_1$,…,*image*$_n$}]**
- **ImageDisplacements - ImageDisplacements[{*image*$_1$,*image*$_2$,…,*image*$_n$}], ImageDisplacements[{*image*$_1$,*image*$_2$,…,*image*$_n$},*flow*]**
- **ImageCorrespondingPoints - ImageCorrespondingPoints[*image*$_1$,*image*$_2$]**
- **FindGeometricTransform - FindGeometricTransform[*pts*$_1$,*pts*$_2$], FindGeometricTransform[*ref*,{*pts*$_1$,*pts*$_2$,…}], FindGeometricTransform[{*pts*$_1$,*pts*$_2$,…}]**
- **ImageDistance - ImageDistance[*image*$_1$,*image*$_2$], ImageDistance[*image*$_1$,*image*$_2$,*pos*], ImageDistance[*image*$_1$,*image*$_2$,*pos*$_1$,*pos*$_2$]**

3.6 Morfologické zpracování obrazu

Kombinace metod z teorie množin, topografie a diskrétní matematiky poskytuje matematické morfologii metody ke zpracování obrazu a jiných diskrétních dat. Wolfram Language zahrnuje rozsáhlou a efektivní implementaci matematické morfologie, zcela integrovanou do jazyku Wolfram Language. V první části jsou popsány funkce zabývající se přípravou obrazu a základní operace pro morfologické zpracování obrazu. Dále zde nalezneme funkce pro morfologické transformace a analýzu komponent.

3.6.1 Příprava obrazu

Funkce **Binarize** a **MorphologicalBinarize** konvertují obraz na černobílo. Funkce **Binarize** určí globální práh, kdy všechny hodnoty nad hodnotou prahu nastaví na hodnotu 1, ostatní na hodnotu 0. Dále je možno zadat zvolenou prahovou hodnotu. Podobně funguje funkce **MorphologicalBinarize.** Ta nahradí všechny hodnoty nad horní prahovou hodnotou 1 včetně hodnot nad dolní prahovou hodnotou, které jsou v popředí. Pokud prahové hodnoty nezadáme, zvolí se automaticky.

In[86]:= `Binarize[Obr]`

Out[86]= 

In[87]:= `MorphologicalBinarize[Obr]`

Out[87]= 

Přehled funkcí:

• **Binarize - Binarize**[*image*], **Binarize**[*image,t*], **Binarize**[*image,*{t_1,t_2}], **Binarize**[*image,f*]

• **MorphologicalBinarize - MorphologicalBinarize**[*image,*{t_1,t_2}], **MorphologicalBinarize**[*image,t*], **MorphologicalBinarize**[*image*]

• **ColorNegate** - viz kapitola 3.4.1

3.6.2 Základní operace

Pomocí funkce **Dilation** získáme morfologickou dilataci obrazu s ohledem na strukturní prvek. Funkci **Dilation** lze dále aplikovat i na pole dat. Funkce **Erosion** rozrušuje obraz s ohledem na strukturní prvek.

In[88]:=
```
Dilation[Obr, 5]
```

Out[88]=

In[89]:=
```
Erosion[Obr, 5]
```

Out[89]=

Pro morfologické otevření obrazu s ohledem na strukturující prvek slouží funkce **Opening**. Naopak pro morfologické uzavření obrazu slouží funkce **Closing**.

In[90]:=
```
Opening[Obr, 5]
```

Out[90]=

In[91]:=
```
Closing[Obr, 5]
```

Out[91]=

Přehled funkcí:

• **Dilation - Dilation**[*image,ker*], **Dilation**[*image,r*], **Dilation**[*data,...*]

• **Erosion - Dilation**[*data,...*], **Erosion**[*image,r*], **Erosion**[*data,...*]

• **Opening - Opening**[*image,ker*], **Opening**[*image,r*], **Opening**[*data,...*]

• **Closing - Closing**[*image,ker*], **Closing**[*image,r*], **Closing**[*data,...*]

3.6.3 Morfologické transformace

Funkce **DistanceTransform** udává vzdálenost transformace obrazu, kde je hodnota každého pixelu nahrazena jeho vzdáleností k nejbližšímu pixelu pozadí. Opakem je funkce **InverseDistanceTransform**, která poskytuje inverzní transformaci vzdálenosti obrazu a jako výsledek vrací binární obraz. **TopHatTransform** a **BottomHatTransform** poskytují morfologickou transformaci obrazu s ohledem na strukturující prvek.

In[92]:=
```
DistanceTransform[ ] // ImageAdjust
```

Out[92]=

In[93]:=
```
InverseDistanceTransform[ ]
```

Out[93]=

In[94]:=
```
TopHatTransform[Obr, 30]
```

Out[94]=

In[95]:=
```
BottomHatTransform[Obr, 30]
```

Out[95]=

MinDetect nám poskytuje binární obraz, ve kterém bílé pixely odpovídají minimu v obraze. Opakem funkce je **MaxDetect**, kde bílé pixely odpovídají maximu v obraze. **FillingTransform** vyplňuje v obraze všechny minima. Mezi další funkce patří **MorphologicalTransform**, což je bloková binární morfologická operace a **MorphologicalGraph**, která vygeneruje obraz z kostry daného obrazu.

In[96]:= `MinDetect[Obr]`

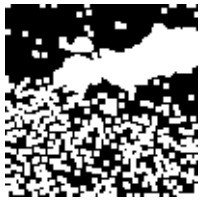
Out[96]=

In[97]:= `MaxDetect[Obr]`

Out[97]=

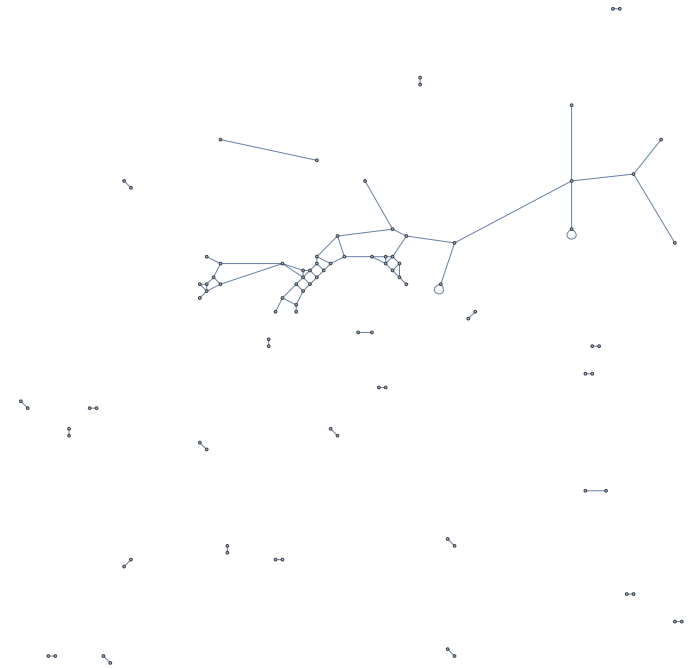
In[98]:= `MorphologicalTransform[`  `, Max]`

Out[98]=



In[99]:= `MorphologicalGraph[`  `, VertexSize -> Large ]`

Out[99]=



Přehled funkcí:

- `DistanceTransform - DistanceTransform[`*image*`], DistanceTransform[`*image,t*`]`
- `InverseDistanceTransfom - InverseDistanceTransform[`*image*`]`
- `TopHatTransform - TopHatTransform[`*image,ker*`], TopHatTransform[`*image,r*`], TopHatTransform[`*data,...*`]`

• **BottonHatTransform - BottomHatTransform**[*image,ker*], **BottomHatTransform**[*image,r*], **BottomHatTransform**[*data,…*]
• **MinDetect - MinDetect**[*image*], **MinDetect**[*image,h*], **MinDetect**[*data,…*]
• **MaxDetect - MaxDetect**[*image*], **MaxDetect**[*image,h*], **MaxDetect**[*data,…*]
• **FillingTransform - FillingTransform**[*image*], **FillingTransform**[*image,marker*], **FillingTransform**[*image,h*]
• **MorphologicalTransform - MorphologicalTransform**[*image,f*], **MorphologicalTransform**[*image,rule*], **MorphologicalTransform**[*image,"name"*], **MorphologicalTransform**[*image,transformation,n*]
• **MorphologicalGraph - MorphologicalGraph**[*image*]

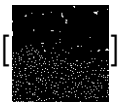
3.6.4 Analýza komponent

Funkce **ComponentMeasurements** provádí analýzu tvarů, barev a vlastností obrazu. **SelectComponents** vybere komponenty obrazu, které splňují kritérium a ostatní části nahradí.

```
In[100]:= ComponentMeasurements[Obr, {"MaxIntensity", "MinIntensity", "MeanIntensity"}]
Out[100]= {1 → {1., 0.0313725, 0.542812}}
```

DeleteSmallComponents nahradí malé komponenty binárního obrazu pixely pozadí.

```
In[101]:= DeleteSmallComponents[
```



```
]
```

Out[101]=

Přehled funkcí:

• **ComponentMeasurements - ComponentMeasurements**[{*image,lmat*},*"prop"*], **ComponentMeasurements**[*image,"prop"*], **ComponentMeasurements**[*…,"prop",crit*], **ComponentMeasurements**[*…,"prop",crit,format*]
• **SelectComponents - SelectComponents**[{*image,lmat*},*crit*], **SelectComponents**[*image,crit*], **SelectComponents**[*…,"prop",n*], **SelectComponents**[*…,"prop",n,p*]
• **DeleteSmallComponents - DeleteSmallComponents**[*image*], **DeleteSmallComponents**[*m*], **DeleteSmallComponents**[*…,n*]
• **Colorize** – viz kapitola 3.4.7
• **HighlightImage** – viz kapitola 3.4.7

3.7 Obrazové filtry a další zpracování obrazu

Wolfram Language obsahuje nejen vysoce optimalizované implementace standardních filtrů pro zpracování obrazu, ale také lze nastavit vlastnosti filtrů. Filtry můžeme rozdělit do několika skupin: lineární filtry, nelineární, frekvenční filtry a další filtry. Vzhledem k množství filtrů je uveden pouze jejich přehled.

3.7.1 Lineární filtry

Jako příklad lineárních filtrů použijeme funkci **MeanFilter**, která filtruje data a nahradí jejich hodnotu střední hodnotou v okolí.

In[102]:=
```
MeanFilter[Obr, 5]
```

Out[102]=

Přehled funkcí:

- **Blur** – viz kapitola 3.3.4
- **Sharpen** – viz kapitola 3.3.3
- Gaussův filtr: **GaussianFilter** – **GaussianFilter**[*data*,*r*], **GaussianFilter**[*data*,*r*,{n_1,n_2,…}], **GaussianFilter**[*data*,{*r*,σ},…], **GaussianFilter**[*data*,{{r_1,r_2,…},…}]
- Gradientní filtr: **GradientFilter** – **GradientFilter**[*data*,*r*], **GradientFilter**[*data*,{*r*,σ}], **GradientFilter**[*data*,{{r_1,r_2,…},…}]
- Filtr orientace gradientu: **GradientOrientationFilter** – **GradientOrientationFilter**[*data*,*r*], **GradientOrientationFilter**[*data*,{*r*,σ}]
- Laplacián Gaussův filtr: **LaplaciantGaussianFilter** – **LaplacianGaussianFilter**[*data*,*r*], **LaplacianGaussianFilter**[*data*,{*r*,σ}]
- Laplacián filtr: **LaplaciantFilter** – **LaplacianFilter**[*data*,*r*], **LaplacianFilter**[*data*,{r_1,r_2,…}]
- **MeanFilter** – **MeanFilter**[*data*,*r*], **MeanFilter**[*data*,{r_1,r_2,…}]
- Wienerův filtr: **WienerFilter** – **WienerFilter**[*data*,*r*], **WienerFilter**[*data*,*r*,*ns*], **WienerFilter**[*data*,{r_1,r_2,…},…]
- **RidgeFilter** – **RidgeFilter**[*data*], **RidgeFilter**[*data*,σ]
- Gáborův filtr: **GaborFilter** – **GaborFilter**[*data*,*r*,*k*], **GaborFilter**[*data*,*r*,*k*,ϕ], **GaborFilter**[*data*,{*r*,σ},…]
- Konvoluce obrazu: **ImageConvolve** – **ImageConvolve**[*image*,*ker*]
- Korelace obrazu: **ImageCorrelate** – **ImageCorrelate**[*image*,*ker*], **ImageCorrelate**[*image*,*ker*,*f*]
- Derivační filtr: **DerivaveFilter** – **DerivativeFilter**[*data*,{n_1,n_2,…}], **DerivativeFilter**[*data*,{n_1,n_2,…},σ], **DerivativeFilter**[*data*,{der_1,der_2,…},…]

3.7.2 Nelineární filtry

Jako příklad nelineárního filtru použijeme funkce **MinFilter** a **MaxFilter**, které nahradí každou hodnotu minimální (maximální) hodnotou v jejím okolí.

In[103]:=
```
MinFilter[Obr, 5]
```

Out[103]=

In[104]:= `MaxFilter[Obr, 6]`

Out[104]=

Přehled funkcí:

- **MedianFilter** – **MedianFilter**[*image*,*r*], **MedianFilter**[*data*,{r_1,r_2,...}]
  Nahradí každou hodnotu střední hodnotou v jejím okolí.
- **MinFilter** – **MinFilter**[*data*,*r*], **MinFilter**[*data*,{r_1,r_2,...}]
  Nahradí každou hodnotu minimální hodnotou v jejím okolí.
- **MaxFilter** – **MaxFilter**[*data*,*r*], **MaxFilter**[*data*,{r_1,r_2,...}]
  Nahradí každou hodnotu maximální hodnotou v jejím okolí.
- **CommonesFilter** – **CommonestFilter**[*data*,*r*], **CommonestFilter**[*data*,{r_1,r_2,...}]
  Nahradí každou hodnotu střední hodnotou v jejím okolí.
- **RangeFilter** – **RangeFilter**[*data*,*r*], **RangeFilter**[*data*,{r_1,r_2,...}]
  Nahradí každou hodnotu rozdílem maxima a minima v jejím okolí.
- **EntropyFilter** – **EntropyFilter**[*data*,*r*], **EntropyFilter**[*data*,{r_1,r_2,...}]
  Nahradí každou hodnotu v jejím okolí entropickou hodnotou.
- **StandardDeviationFilter** – **StandardDeviationFilter**[*data*,*r*], **StandardDeviationFilter**[*data*,{r_1,r_2,...}]
  Nahradí každou hodnotu směrodatnými odchylkami hodnot v jejím okolí.
- **HarmonicMeanFilter** – **HarmonicMeanFilter**[*data*,*r*], **HarmonicMeanFilter**[*data*,{r_1,r_2,...}]
  Nahradí každou hodnotu harmonickou průměrnou hodnotou v jejím okolí.
- **GeometricMeanFilter** – **GeometricMeanFilter**[*data*,*r*], **GeometricMeanFilter**[*data*,{r_1,r_2,...}]
  Nahradí každou hodnotu geometricky průměrnou hodnotou v jejím okolí.
- Kuwaharův filtr: **KuwaharaFilter** – **KuwaharaFilter**[*data*,*r*]
- **MeanShiftFilter** – **MeanShiftFilter**[*data*,*r*,*d*], **MeanShiftFilter**[*data*,{r_1,r_2,...},*d*]
  Nahradí každou hodnotu průměrem hodnot v jejím okolí ve zvolené vzdálenosti.
- Perona-Malik filtr: **PeronaMalikFilter** – **PeronaMalikFilter**[*image*], **PeronaMalikFilter**[*image*,*t*], **PeronaMalikFilter**[*image*,*t*,*k*], **PeronaMalikFilter**[*image*,*t*,*k*,σ]
- **CurvatureFlowFilter** – **CurvatureFlowFilter**[*image*], **CurvatureFlowFilter**[*image*,*t*], **CurvatureFlowFilter**[*image*,*t*,*k*]
  Aplikuje na obraz filtr zakřivení.

3.7.3 Frekvenční filtry

Jako příklad pro frekvenční filtry použijeme funkci **LowpassFilter**, která aplikuje dolní propust s mezní frekvencí ω_c na pole dat.

In[105]:= `LowpassFilter[Obr, 0.5]`

Out[105]=

Přehled funkcí:

- Dolní propust: **LowpassFilter - LowpassFilter**[*data*,ω_c], **LowpassFilter**[*data*,ω_c,*n*], **LowpassFilter**[*data*,ω_c,*n*,*wfun*]
- Horní propust: **HighpassFilter - HighpassFilter**[*data*,ω_c], **HighpassFilter**[*data*,ω_c,*n*], **HighpassFilter**[*data*,ω_c,*n*,*wfun*]
- Pásmová propust: **BandpassFilter - BandpassFilter**[*data*,{ω_1,ω_2}], **BandpassFilter**[*data*,{{ω,*q*}}], **BandpassFilter**[**data**,**spec**,**n**], **BandpassFilter**[*data*,*spec*,*n*,*wfun*]
- Pásmová zádrž: **BandstopFilter - BandstopFilter**[*data*,{ω_1,ω_2}], **BandstopFilter**[*data*,{{ω,*q*}}], **BandstopFilter**[**data**,**spec**,**n**], **BandstopFilter**[*data*,*spec*,*n*,*wfun*]
- Derivační článek: **DifferentiatorFilter - DifferentiatorFilter**[*data*,ω_c], **DifferentiatorFilter**[*data*,ω_c,*n*], **DifferentiatorFilter**[*data*,ω_c,*n*,*wfun*]
- Hilbertův filtr: **HilbertFilter - HilbertFilter**[*data*,ω_c], **HilbertFilter**[*data*,ω_c,*n*], **HilbertFilter**[*data*,ω_c,*n*,*wfun*]

3.7.4 Další funkce pro práci s filtry

Masking je možnost pro zpracování obrazu a signálu. **ImageFilter** aplikuje zvolenou funkci na každý pixel v každém kanále obrazu. Dekonvoluci obrazu poskytuje funkce **ImageDeconvolve**. **TotalVariationFilter** slouží ke snížení šumu v obrázku.

In[106]:= `ImageFilter[Max, Obr, 5]`

Out[106]=

Přehled funkcí:

- **Masking**
- **ImageFilter - ImageFilter**[*f*,*image*,*r*]
- **ImageDeconvolve - ImageDeconvolve**[*image*,*ker*]
- **TotalVariationFilter - TotalVariationFilter**[*data*], **TotalVariationFilter**[*data*,*param*]

3.8 Detekce

Wolfram Language poskytuje funkce pro detekci a extrahování prvků v obrazech. Podporuje například specifické geometrické prvky a klíčové body.

3.8.1 Detekce zájmových bodů

ImageKeypoints vyhledá klíčové rysy obrázku a vrátí jejich souřadnici. **Lenght** v příkladu nám vrací počet klíčových rysů.

In[107]:= `ImageKeypoints[Obr] // Length`

Out[107]= 24

ImageCorners detekuje na obrázku hrany a vrací jejich souřadnice. **CornerFilter** vypočítá míru přítomnosti hrany pro každý pixel a vrátí jako výsledek obraz intenzity.

Přehled funkcí:

- `ImageKeypoints - ImageKeypoints[image],ImageKeypoints[image,prop]`
- `ImageCorners - ImageCorners[image], ImageCorners[image,r], ImageCorners[image,r,t], ImageCorners[image,r,t,d]`
- `CornerFilter - CornerFilter[image], CornerFilter[image,r]`

3.8.2 Detekce obrysů

Detekce hran pomocí Canny metody a dalších metod provádí funkce **EdgeDetect**. Ta nalezne hrany v obraze a vrátí jako výsledek binární obraz. Dalšími funkcemi je **CrossingDetect** a **ContourDetect.** Tyto funkce detekují nulové hodnoty, které následně odpovídají bílým pixelům ve výsledném binárním obraze.

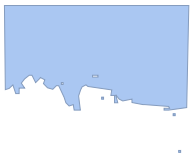
In[108]:= `EdgeDetect[Obr]`

Out[108]=

ImageLines slouží k detekci přímek v obraze a vrátí souřadnice jejich bodů. K vrácení popředí obrázku se využívá funkce **ImageMesh**. Pro oddělení pozadí lze využít funkci **RemoveBackground**, která vrátí obraz, který obsahuje alfa kanál a na kterém je pozadí průhledné.

In[109]:= `ImageMesh[Obr, ImageSize → Tiny]`

Out[109]=



In[110]:= `RemoveBackground[Obr]`

Out[110]=

Přehled funkcí:

- `EdgeDetect - EdgeDetect[image], EdgeDetect[image,r], EdgeDetect[image,r,t]`
- `CrossingDetect - CrossingDetect[image], CrossingDetect[image,delta], CrossingDetect[array,...]`
- `ContourDetect - ContourDetect[image], ContourDetect[image,delta], ContourDetect[array,...]`
- `ImageLines - ImageLines[image], ImageLines[image,t], ImageLines[image,t,d]`
- `ImageMesh - ImageMesh[image]`
- `GradientFilter` - viz kapitola 3.7.1

- **LaplacianGaussianFilter** – viz kapitola 3.7.1
- **ImageFilter** – viz kapitola 3.7.4
- **ImageConvolve** – viz kapitola 3.7.1
- **RemoveBackground** – **RemoveBackground**[*image*], **RemoveBackground**[*image*,*model*]

3.9 Počítačové vidění

Wolfram Langueage obsahuje funkce pro identifikaci obrazu, detekci a rozpoznávání objektů. V této kapitole je popsán přehled funkcí, využití vybraných funkcí je popsáno v kapitole 4.2.

3.9.1 Detekce objektů

Pro detekci objektů lze využít celou řadu funkcí. **ImageCases** vyhledá objekty na obraze, jako výstup poskytne seznam všech kategorií vyskutujících se na obraze. **FindFaces** vyhledá na obrázku lidské tváře. Vyhodnocení je buď ve formě souřadnice pro nalezenou tvář nebo lze zadat specifické vlastnosti, jako například zvýraznění tváře na původním obrázku či výřez tváře. Další funkcí je **FacialFeatures**, která detekuje rysy obličeje a vrací seznam detekovaných osob, jejich tvář, odhad věku, pohlaví a emoce. Dále lze rozšířit o argumenty vracející pozici např. pravého/levého oka a obočí, nosu, úst, obrys obličeje.

ImagePosition vrací pozici pro každou identifikovanou kategorii objektů na obrázku. K ohraničení identifikovaných objektů na obrázku slouží funkce **ImageBoundingBoxes**. K testování, zda na obrázku je daná kategorie objektů se využívá funkce **ImageContainsQ**, která vrací hodnotu True, v ostatních případech vrací hodnotu False.

Přehled funkcí:

- **ImageCases** – **ImageCases**[*image*], **ImageCases**[*image*,*category*], **ImageCases**[*image*,*category*→*prop*], **ImageCases**[*image*,{*category*$_1$,*category*$_2$,…}]
- **FindFaces** – **FindFaces**[*image*], **FindFaces**[*image*,*prop*], **FindFaces**[*image*,*crit*,*prop*]
- **FacialFeatures** – **FacialFeatures**[*image*], **FacialFeatures**[*image*,*features*]
- **RemoveBackground** – viz kapitola 3.8.2
- **ImagePosition** – **ImagePosition**[*image*], **ImagePosition**[*image*,*obj*]
- **ImageBoundingBoxes** – **ImageBoundingBoxes**[*image*], **ImageBoundingBoxes**[*image*,*category*]
- **ImageContainsQ** – **ImageContainsQ**[image,category], **ImageContainsQ**[*image*,{*category*$_1$,*category*$_2$,…}], **ImageContainsQ**[*image*,*category*$_1$|*category*$_2$|…]

3.9.2 Rozpoznávání objektů

Funkce **ImageIdentify** se využívá k identifikaci objektů na obrázku či k další podrobnější specifikaci objektů. Zda-li se na obrázku nachází zvolený objekt, testuje funkce **ImageInstanceQ**, která vrací hodnotu True, jestliže se na obrázku daný objekt vyskytuje, v ostatním případě vrací hodnotu False. **ImageContents** identifikuje objekty na obrázku, jeho výstupem je obrazový přehled identifikovaných objektů, kategorie, souřadnice ohraničení daného objektu a pravděpodobnost identifikovaného objektu.

In[111]:=
```
ImageIdentify[Obr]
```

Out[111]= observatory

K rozeznávání textu na obrázku se využívá funkce **TextRecognize**, která vrací rozpoznaný text jako řetězec znaků. Mezi podporovanými jazyky je i čeština. Obdobou je **BarcodeRecognize** sloužící k rozpoznání čárového či QR kódu.

Přehled funkcí:

- **ImageIdentify - ImageIdentify**[*image*], **ImageIdentify**[*image,category*], **ImageIdentify**[*image,category,n*], **ImageIdentify**[*image,category,n,”prop”*]
- **ImageInstanceQ - ImageInstanceQ**[*image,obj*], **ImageInstanceQ**[*image,*obj_1/obj_2/*...*], **ImageInstanceQ**[*image,obj,category*]
- **ImageContents - ImageContents**[*image*], **ImageContents**[*image,category*], **ImageContents**[*image,category,prop*]
- **TextRecognize - TextRecognize**[*image*], **TextRecognize**[*image,level*], **TextRecognize**[*image,level,prop*]
- **BarcodeRecognize - BarcodeRecognize**[*image*], **BarcodeRecognize**[*image,”prop”*], **BarcodeRecognize**[*image,”prop”,format*]

4 Využití funkcí v praxi

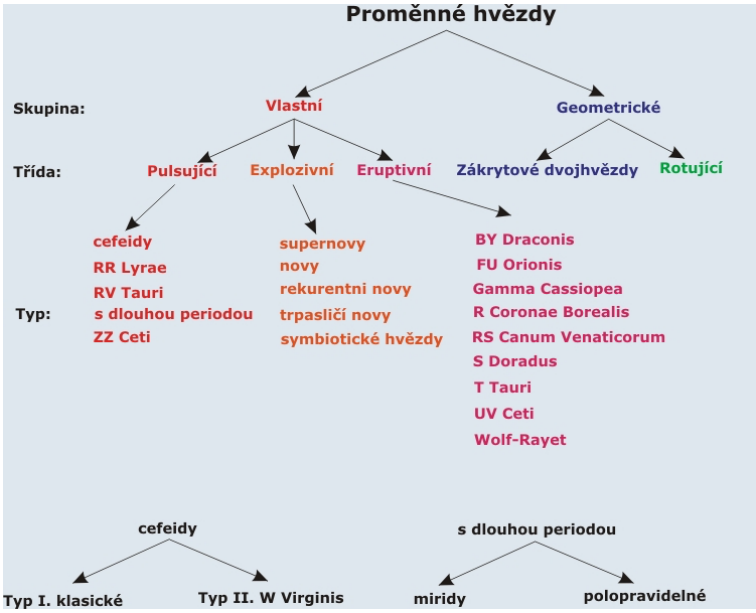
4.1 Proměnné hvězdy

První pozorování proměnných hvězd bylo založeno na vizuální fotometrii. Jako první jasně formulovanou pozorovací metodu definoval F. W. Herschel, který srovnával jasnosti dvou hvězd a zapisoval jej pomocí značek a symbolů. Ovšem F. W. A. Argelander si uvědomil nedostatky Herschelovy metody a následně zavedl místo značek jasně definované stupně s číselným vyjádřením. Další pozorovací metoda byla Pogsonova, která je založena na znalosti hvězdných velikostí srovnávacích hvězd. Koncem 19. století popsal americký astronom E. C. Pickering interpolační metodu využívající vždy dvojici srovnávacích hvězd, které mají známou hvězdnou velikost. Poslední vizuální pozorovací metoda byla kombinací Argelanderovy stupňové a Pickeringovy interpolační metody, kterou navrhli A. A. Nijland a S. N. Blažko nezávisle na sobě [23].

V současné době se již používá při pozorovaní CCD (Charge Couple Device) technika, která pozorování značně zjednodušila. Počátek nového věku astronomie nastal v roce 1979, kdy byl jako první v astronomii použit CCD prvek, a to na dalekohledu na Kitt Peak National Observatory. Mezi značné výhody patří vysoká kvantová účinnost, lineární odezva čipů, lze sledovat velmi slabé hvězdy, citlivost v červené a v současnosti i v modré oblasti spektra. Mezi nevýhody se řadí menší dynamický rozsah a citlivost [23].

Proměnné hvězdy jsou hvězdy, u kterých se mění jasnost v čase. Mechanismy proměnných hvězd můžeme rozdělit na geometrické nebo fyzické. Rozdělení proměnných hvězd na obr. 2.

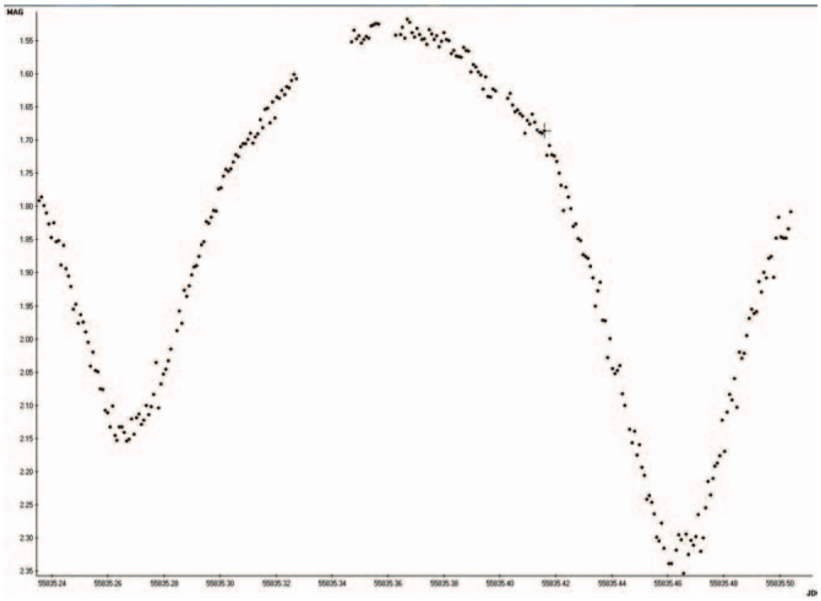
- **Geometrické mechanismy** – mezi ně patří zákrytové hvězdy, což jsou zpravidla dvojhvězdy, které obíhají okolo společného těžiště a pokud rovina tohoto oběhu leží ve směru našeho pozorování, pozorujeme vzájemné zákryty hvězd. Dále se mezi geometrické mechanismy řadí rotující hvězdy, na jejichž povrchu jsou chladnější místa, z kterých přichází méně fotonů a vlivem rotace hvězdy kolem vlastní osy pozorujeme teplotně různé oblasti.
- **Fyzické mechanismy** – jde o vlastní proměnné hvězdy, u kterých se mění např. radiální rychlost, povrchová teplota nebo spektrum. Patří sem pulsující proměnné hvězdy, eruptivní proměnné hvězdy a explozivní proměnné hvězdy [24], [25].



Obr. 2: *Rozdělení proměnných hvězd. Převzato z [24].*

Fotometrie se zabývá měřením toku a intenzity světla v oboru elektromagnetického spektra. Hvězdy považujeme za zdroje elektromagnetického záření, které do prostoru vyzařují izotropně a jejich celkový zářivý výkon (zářivý tok) odpovídá celkové energii vyzářené ve všech vlnových délkách za jednotku času. Jasnost zdroje záření je popisována tzv. hvězdnou velikostí neboli magnitudou. Hvězdná velikost je logaritmická veličina [26].

Hlavním zdrojem informací o proměnných hvězdách je světelná křivka, což je závislost jasnosti na čase, kde jasnost je proměnná veličina. Na křivce lze pozorovat sekundární a primární minimum, které je znázorněné na obr. 3 [26].



Obr. 3: *Světelná křivka EW Del, sekundární a primární minimum. Převzato z [26].*

Pozorování proměnných hvězd závisí na několika parametrech, mezi které např. patří propustnost filtrů, kvantová účinnost optiky, kvantová účinnost detektoru, propustnost atmosféry a kvalita pozorovacích podmínek, mezi které řadíme tzv. seeing, oblačnost a vzdušnou hmotu. Mezi další vliv řadíme zeslabení světla hvězdy působením mezihvězdné látky [26].

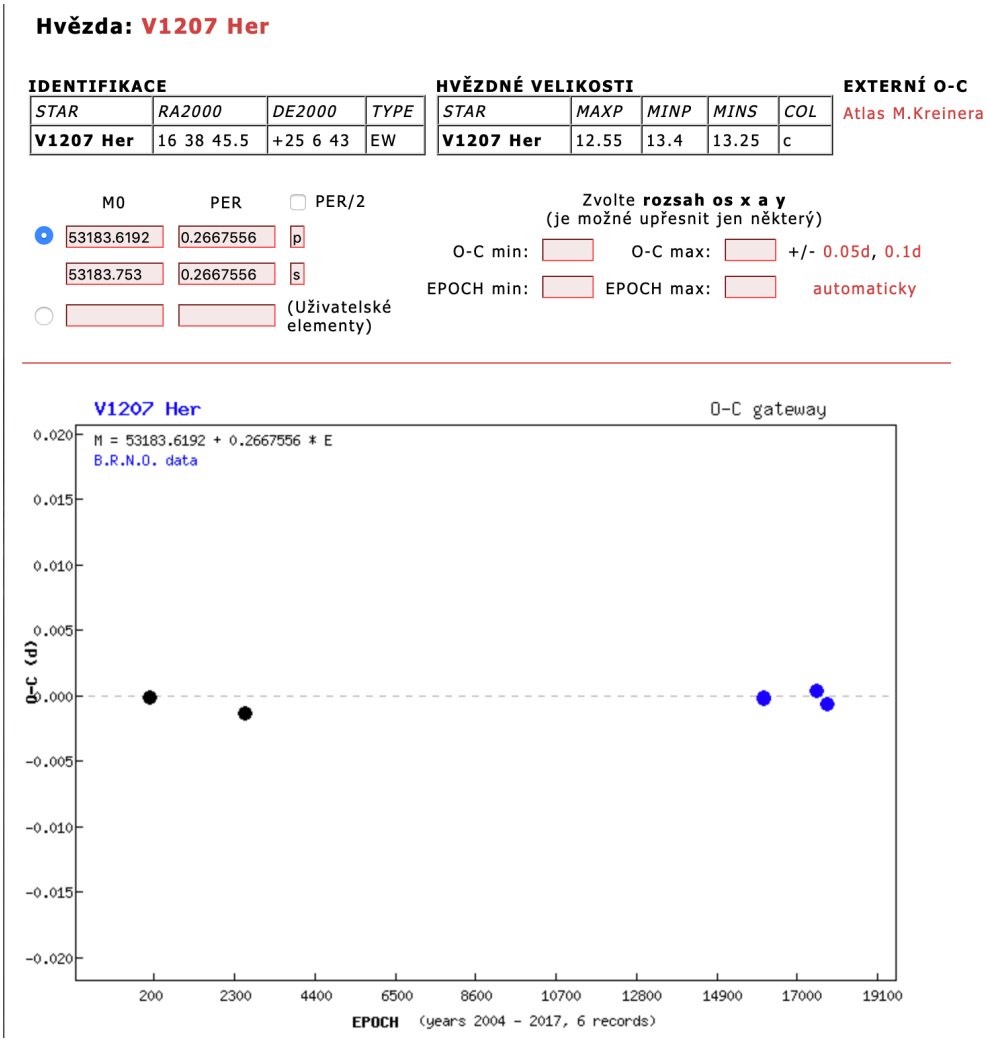
Pozorování nám ve značné míře nejvíce ovlivňují turbulence v atmosféře. Pozorovaný předmět při velmi krátké expoziční době se velmi rychle mění, naopak při dlouhém expozičním čase je rozmazaný. Charakterizovat turbulence v atmosféře můžeme pomocí prostorových vlastností (tzv. seeing – velikost difrakčního obrazce, která je dána vlivem turbulencí), časových vlastností (speklový dehorenční čas) a izoplanatického úhlu (maximální úhlová vzdálenost dvou objektů, kde zkreslení můžeme považovat za totožné) [27].

CCD čip je založen na fotoelektrickém jevu, kde přicházející světlo vytváří v polovodiči elektrický náboj. Ovšem elektrony se nemohou kvůli negativně potenciálovým valům volně pohybovat po čipu. Dále je zde systém vodorovných elektrod, který má negativní náboj a tvoří na čipu mřížku tzv. potenciálových studní. Každý pixel je tvořen jednou potenciálovou studní. Výhodnou CCD čipů je schopnost akumulace náboje po dlouhou dobu. Tedy pokud dopadne na pixel větší množství světla, naakumulují více elektronů. Značnou nevýhodou je ovšem možnost "přelévání" elektronů do sousední nábojové studně a nastává tzv. blooming. Tento jev se vyskytuje v případě velmi jasných hvězd. Následně balíky elektronů se posouvají do výstupního zesilovače a dochází k převodu elektrického náboje na napětí, který změří elektronika kamery zvlášť pro každý pixel. Další nevýhodou CCD čipů je tzv. temný proud, který závisí na okolní teplotě, proto je nutné kameru chladit [26].

K odstranění temného šumu se používá tzv. temný snímek (dark frame), který se provádí při expozici s uzavřenou uzávěrkou. Jestliže se oba snímky odečtou, lze odstranit obraz vygenerovaný temným proudem. K odstranění multiplikativního šumu a nerovnoměrného osvětlení detektoru se využívá podělení snímku tzv. flat-fieldem (master-flat), který se pořizuje za soumraku nebo za úsvitu, kdy není obloha příliš jasná a nehrozí tak přesvětlení detektoru [26].

4.1.1 Proměnná hvězda V1207 Her

Pozorovaná proměnná hvězda V1207 Her je typu EW, což znamená zákrytovou hvězdu typu W Ursae Majoris, což jsou velmi málo hmotné dotykové dvojhvězdy. Typ W označuje chladnější hvězdy spektrální třídy G nebo K, jejichž perioda se pohybuje v rozmezí 0,22 - 0,4 dne [28]. Další informace jsou k dispozici na Sekci pozorovatelů proměnných hvězd České astronomické společnosti. Na obr. 4 můžeme vidět změnu hvězdné velikosti, při primárním minimu je 13,4 mag, při sekundárním 13,25 mag. Dále je zde perioda primárního a sekundárního minima proměnné hvězdy a její pozorování [29].



Obr. 4: *Proměnná hvězda V1207 Her. Převzato z [29].*

Měření bylo prováděno 22. 8. 2018 na hvězdárně ve Valašském Meziříčí, s použitím přístroje Newton 254/1200-G2 402. Výsledné snímky byly zpracovány a opraveny v programu *Muniwin* o master-dark a master-flat snímek. Zpracovávané snímky byly pořízeny s filtrem B (vlnová délka 440 nm), délka expozice 60 s, chlazení CCD čipu na -30 °C.

4.1.2 Nalezení nejjasnějších hvězd na snímku

Pro nalezení hvězd na snímku nejprve importujeme obraz dané hvězdy ve tvaru fits. Formát fits vychází z anglického Flexible Image Transport System, který slouží pro ukládání astronomických dat. Jde o formát grafického souboru, ale obsahuje i informace o metadatách jako např. fotometrické informace [30].

Nejprve vložíme snímek hvězdy pomocí funkce **Import**, následně i snímky Masterdark a Masterflat. Při nahrání snímku do programu Wolfram Mathematica dostáváme pouze grafický soubor.

```
In[112]:= ImportV1207 = Import["V1207 Her.fits"]
```

Out[112]= ⟨| 1 →  |⟩

```
In[113]:= ImportMasterdark = Import["masterdark.fits"]
```

Out[113]= ⟨| 1 →  |⟩

```
In[114]:= ImportMasterflat = Import["masterflat.fits"]
```

Out[114]= ⟨| 1 →  |⟩

Neprve provedeme opravení V1207 snímku o Masterdark pomocí funkce **ImageSubtract**, která daný snímek Masterdark odečte od původního V1207. V dalším kroku provedeme opravu o Masterflat a to pomocí funkce **ImageDifference**, která poskytne obraz, kde je každý pixel absolutním rozdílem odpovídajících pixelů v jednotlivých obrazech.

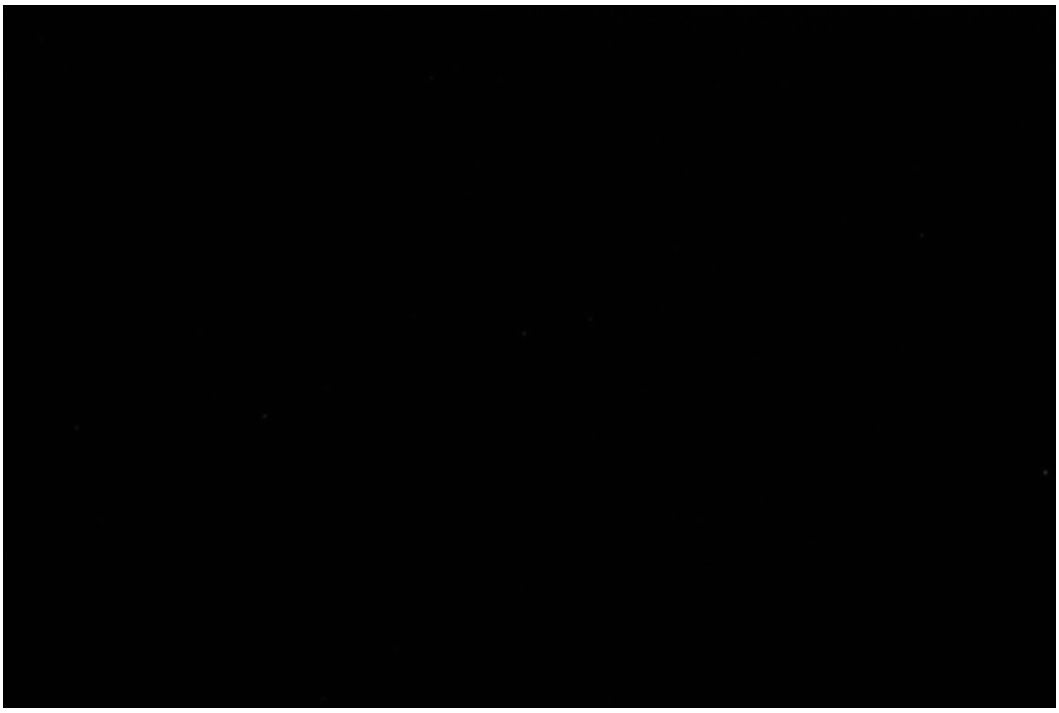
```
In[115]:= V1207 =  ;
```

```
Masterdark =  ;
```

```
Masterflat =  ;
```

In[116]:= `V1207Dark = ImageSubtract[V1207, Masterdark]`

Out[116]=



In[117]:= `V1207Flat = ImageDifference[ Masterflat, V1207Dark]`

Out[117]=

Na takto upravený snímek “V1207Flat” pro zvýraznění hvězd nejprve provedeme negaci barev pomocí **ColorNegate** a v posledním kroku použijeme funkci **Binarize**, která převede obraz do binárních hodnot a v argumentu lze měnit globální práh a tím i jak výrazné hvězdy chceme na snímku zobrazit.

In[118]:=
```
V1207Neg = ColorNegate[V1207Flat]
```

Out[118]=

In[119]:=
```
V1207Bin = Binarize[V1207Neg, 0.865]
```

Out[119]=

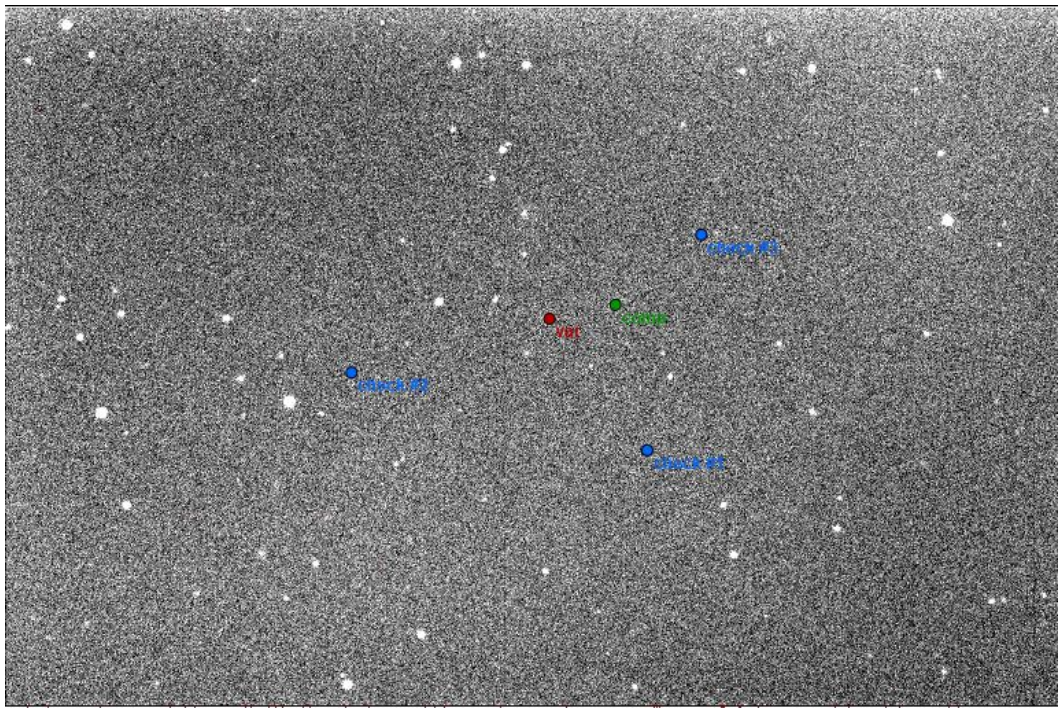
Na výsledním snímku “V1207Bin” lze pozorovat nejvýraznější hvězdy na pořízeném snímku. Ve středu snímku se nachází pozorovaná proměnná hvězda V1207 Her.

4.1.3 Měření poloměru proměnné a srovnávací hvězdy

V této části se budeme zabývat změnou poloměru proměnné hvězdy se srovnávací hvězdou v čase. Srovnávací hvězda by měla mít přibližně stejnou magnitudu jakou má proměnná hvězda. Nejprve vložíme obrázek, na kterém je červeně vyznačena poloha proměnné hvězdy a zeleně srovnávací hvězdy.

In[120]:=
```
Poloha = Import["V1207 Poloha.jpg"]
```

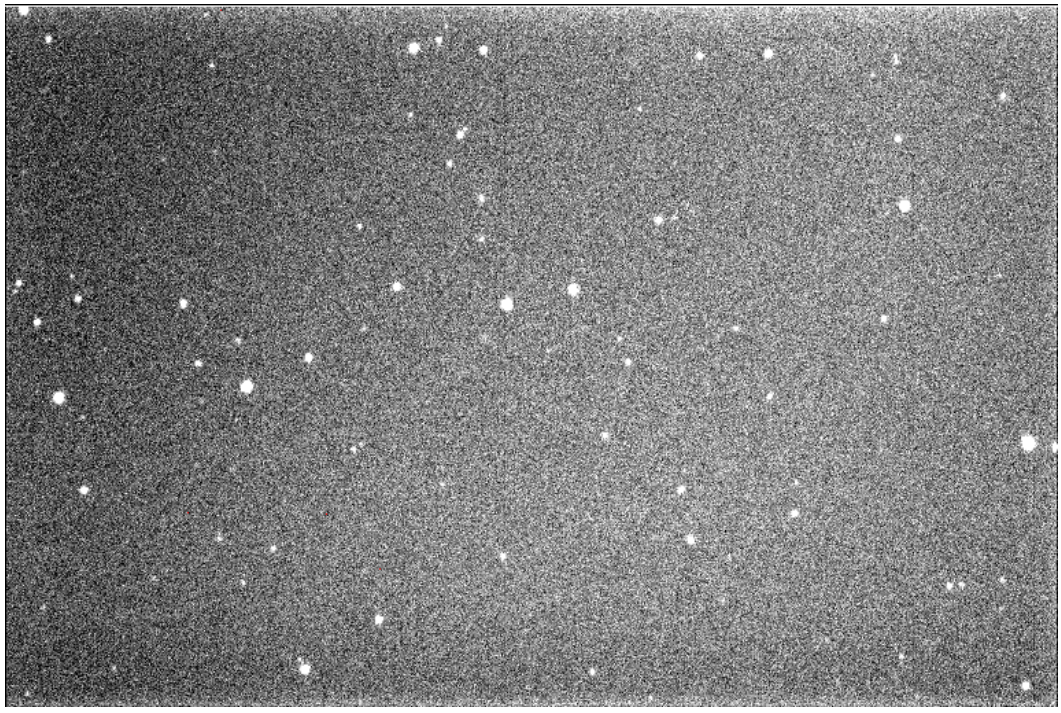
Out[120]=



Jako první vložíme vybrané snímky na zpracování, které jsou označené “her*” (hvězdička * značí pořadí vybraného snímku z měření). Tento surový snímek byl v programu *Muniwin* již opraven o masterdark a masterflat snímek. Jelikož se nepodařilo při pozorování udržet pozici proměnné hvězdy na jednom místě, musíme každý vybraný snímek zpracovat zvlášť. Ořezání snímku na oblast proměnné hvězdy provedeme funkcí **ImageTake**. Takto ořezanou proměnnou hvězdu pojmenujeme “var*”.

In[121]:=
```
her10 = Import["frame10.png"]
```

Out[121]=



In[122]:=
```
var10 = ImageTake[her10, {200, 250}, {350, 400}]
```

Out[122]=

Filtr **TotalVariationFilter** nám pomůže se zbavit nežádoucího šumu. Hodnotu v druhém argumentu funkce (efektivnost filtrace), nastavíme tak, aby se nám nežádoucí šum slil do šedého pozadí, ale zároveň aby stále šlo detekovat proměnnou hvězdu.

In[123]:=
```
filtr10 = TotalVariationFilter[var10, 0.3]
```

Out[123]=

V dalším kroku využijeme funkci **Binarize**, která nám ještě více pomůže redukovat šum. Daný výřez převede na snímek, na kterém budou pixely pouze o dvou hodnotách.

In[124]:=
```
bin10 = Binarize[filtr10]
```

Out[124]=

Následně budeme hledat velikost hvězdy. K tomu použijeme funkci **ComponentMeasurements**. Pomocí argumetů **Centroid** a **EquivalentDiskRadius** budeme hledat pouze rovnoměrně kruhové objekty. Jako výsledek "sou*" dostaneme souřadnice s číslem objektu, jeho souřadnice a poloměr.

In[125]:=
```
sou10 = ComponentMeasurements[bin10, {"Centroid", "EquivalentDiskRadius"}]
```

Out[125]=
```
{1 → {{17.1264, 31.533}, 5.38203}}
```

V posledním kroku na daném výřezu znázorníme červeně proměnnou hvězdu "show*".

In[126]:=
```
show10 = Show[var10, Graphics[{Red, Thick, Circle @@@ sou10[[All, 2]]}]]
```

Out[126]=

Tytéž kroky budeme aplikovat i na ostatní vybrané snímky.

In[127]:=
```
her20 = Import["frame20.png"];
```

In[128]:=
```
var20 = ImageTake[her20, {220, 270}, {360, 410}];
```

In[129]:=
```
sou20 = ComponentMeasurements[Binarize[TotalVariationFilter[var20, 0.3]],
    {"Centroid", "EquivalentDiskRadius"}];
```

In[130]:=
```
show20 = Show[var20, Graphics[{Red, Thick, Circle @@@ sou20[[All, 2]]}]];
```

In[131]:=
```
her30 = Import["frame30.png"];
```

In[132]:=
```
var30 = ImageTake[her30, {200, 250}, {370, 420}];
```

In[133]:=
```
sou30 = ComponentMeasurements[Binarize[TotalVariationFilter[var30, 0.3]],
    {"Centroid", "EquivalentDiskRadius"}];
```

In[134]:=
```
show30 = Show[var30, Graphics[{Red, Thick, Circle @@@ sou30[[All, 2]]}]];
```

In[135]:=
```
her40 = Import["frame40.png"];
```

```
In[136]:= var40 = ImageTake[her40, {215, 265}, {360, 410}];

In[137]:= sou40 = ComponentMeasurements[Binarize[TotalVariationFilter[var40, 0.3]],
            {"Centroid", "EquivalentDiskRadius"}];

In[138]:= show40 = Show[var40, Graphics[{Red, Thick, Circle @@@ sou40[[All, 2]]}]];

In[139]:= her50 = Import["frame50.png"];

In[140]:= var50 = ImageTake[her50, {230, 270}, {380, 430}];

In[141]:= sou50 = ComponentMeasurements[Binarize[TotalVariationFilter[var50, 0.4]],
            {"Centroid", "EquivalentDiskRadius"}];

In[142]:= show50 = Show[var50, Graphics[{Red, Thick, Circle @@@ sou50[[All, 2]]}]];

In[143]:= her60 = Import["frame60.png"];

In[144]:= var60 = ImageTake[her60, {210, 260}, {400, 450}];

In[145]:= sou60 = ComponentMeasurements[Binarize[TotalVariationFilter[var60, 0.3]],
            {"Centroid", "EquivalentDiskRadius"}];

In[146]:= show60 = Show[var60, Graphics[{Red, Thick, Circle @@@ sou60[[All, 2]]}]];

In[147]:= her70 = Import["frame70.png"];

In[148]:= var70 = ImageTake[her70, {240, 290}, {380, 430}];

In[149]:= sou70 = ComponentMeasurements[Binarize[TotalVariationFilter[var70, 0.3]],
            {"Centroid", "EquivalentDiskRadius"}];

In[150]:= show70 = Show[var70, Graphics[{Red, Thick, Circle @@@ sou70[[All, 2]]}]];
```

U srovnávací hvězdy postupujeme stejně jako u proměnné s tím rozdílem, že výslednou hvězdu zvýrazníme modře.

```
In[151]:= comp10 = ImageTake[her10, {180, 230}, {390, 440}];

In[152]:= compsou10 = ComponentMeasurements[Binarize[TotalVariationFilter[comp10, 0.3]],
            {"Centroid", "EquivalentDiskRadius"}];

In[153]:= compshow10 =
            Show[comp10, Graphics[{Blue, Thick, Circle @@@ compsou10[[All, 2]]}]];

In[154]:= comp20 = ImageTake[her20, {200, 250}, {400, 450}];

In[155]:= compsou20 = ComponentMeasurements[Binarize[TotalVariationFilter[comp20, 0.3]],
            {"Centroid", "EquivalentDiskRadius"}];

In[156]:= compshow20 =
            Show[comp20, Graphics[{Blue, Thick, Circle @@@ compsou20[[All, 2]]}]];

In[157]:= comp30 = ImageTake[her30, {190, 240}, {420, 470}];

In[158]:= compsou30 = ComponentMeasurements[Binarize[TotalVariationFilter[comp30, 0.3]],
            {"Centroid", "EquivalentDiskRadius"}];

In[159]:= compshow30 =
            Show[comp30, Graphics[{Blue, Thick, Circle @@@ compsou30[[All, 2]]}]];
```

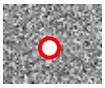
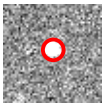
```
In[160]:= comp40 = ImageTake[her40, {210, 260}, {410, 460}];

In[161]:= compsou40 = ComponentMeasurements[Binarize[TotalVariationFilter[comp40, 0.3]],
            {"Centroid", "EquivalentDiskRadius"}];

In[162]:= compshow40 =
            Show[comp40, Graphics[{Blue, Thick, Circle @@@ compsou40[[All, 2]]}]];

In[163]:= comp50 = ImageTake[her50, {220, 270}, {420, 470}];

In[164]:= compsou50 = ComponentMeasurements[Binarize[TotalVariationFilter[comp50, 0.3]],
            {"Centroid", "EquivalentDiskRadius"}];

In[165]:= compshow50 =
            Show[comp50, Graphics[{Blue, Thick, Circle @@@ compsou50[[All, 2]]}]];

In[166]:= comp60 = ImageTake[her60, {200, 250}, {450, 500}];

In[167]:= compsou60 = ComponentMeasurements[Binarize[TotalVariationFilter[comp60, 0.3]],
            {"Centroid", "EquivalentDiskRadius"}];

In[168]:= compshow60 =
            Show[comp60, Graphics[{Blue, Thick, Circle @@@ compsou60[[All, 2]]}]];

In[169]:= comp70 = ImageTake[her70, {230, 280}, {430, 480}];

In[170]:= compsou70 = ComponentMeasurements[Binarize[TotalVariationFilter[comp70, 0.3]],
            {"Centroid", "EquivalentDiskRadius"}];

In[171]:= compshow70 =
            Show[comp70, Graphics[{Blue, Thick, Circle @@@ compsou70[[All, 2]]}]];
```

Z výsledných hodnot sestavíme tabulky. Čas pořízení snímku uvádíme v Juliánském datu (JD). Hodnoty hvězdné velikosti jsou získané z programu *Muniwin*.

Tabulka pro proměnnou hvězdu:

In[172]:=
```
vartab = TableForm[{{"Pořadí snímku", "Snímek", "Čas [JD]", "Poloměr [px]", "Hvězdná velikost [mag]"},
   {10, show10, "2458353.2964", 5.382025607773058, 13.26},
   {20, show20, "2458353.3109", 4.918490759365935, 13.35},
   {30, show30, "2458353.3254", 4.853342309955121, 13.61},
   {40, show40, "2458353.3398", 4.54864184146723, 13.91},
   {50, show50, "2458353.3543", 5.014626706796775, 13.72},
   {60, show60, "2458353.3688", 5.382025607773058, 13.50},
   {70, show70, "2458353.3833", 5.641895835477563, 13.40}}]
```

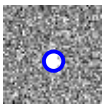
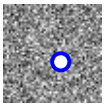
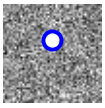
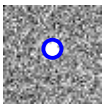
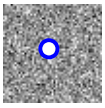
Out[172]//TableForm=

Pořadí snímku	Snímek	Čas [JD]	Poloměr [px]	Hvězdná velikost [mag]
10		2458353.2964	5.38203	13.26
20		2458353.3109	4.91849	13.35
30		2458353.3254	4.85334	13.61
40		2458353.3398	4.54864	13.91
50		2458353.3543	5.01463	13.72
60		2458353.3688	5.38203	13.5
70		2458353.3833	5.6419	13.4

Tabulka pro srovnávací hvězdu:

In[173]:=
```
Comptab = TableForm[{{"Pořadí
snímku", "Snímek", "Čas [JD]", "Poloměr [px]", "Hvězdná velikost [mag]"},
   {10, compshow10, "2458353.2964", 5.046265044040321, 13.99},
   {20, compshow20, "2458353.3109", 4.478115991081385, 14.02},
   {30, compshow30, "2458353.3254", 4.720348719413148, 14.02},
   {40, compshow40, "2458353.3398", 4.720348719413148, 14.02},
   {50, compshow50, "2458353.3543", 4.686510657907603, 14.07},
   {60, compshow60, "2458353.3688", 4.478115991081385, 14.09},
   {70, compshow70, "2458353.3833", 4.720348719413148, 14.10}}]
```

Out[173]//TableForm=

Pořadí snímku	Snímek	Čas [JD]	Poloměr [px]	Hvězdná velikost [mag]
10		2458353.2964	5.04627	13.99
20		2458353.3109	4.47812	14.02
30		2458353.3254	4.72035	14.02
40		2458353.3398	4.72035	14.02
50		2458353.3543	4.68651	14.07
60		2458353.3688	4.47812	14.09
70		2458353.3833	4.72035	14.1

Dále sestavíme graf změny poloměru na čase pro proměnnou hvězdu.Hodnoty pojmenované jako “Varradius”, graf “Vargraf”. Graf sestavíme i pro porovnávací hvězdu. Hodnoty pojmenované jako “Compradius”, graf “Compgraf”.

In[174]:=
```
Varradius = {{0.2964, 5.382025607773058},
   {0.3109, 4.918490759365935}, {0.3254, 4.853342309955121},
   {0.3398, 4.54864184146723}, {0.3543, 5.014626706796775},
   {0.3688, 5.382025607773058}, {0.3833, 5.641895835477563}};
```

In[175]:=
```
Vargraf = Show[ListPlot[Varradius, Joined → True,
    PlotLegends → {"Ilustrační lineární spojnice poloměrů"},
    PlotStyle → LightRed, AxesLabel → {"2458353 JD", "Poloměr [px]"}],
   ListPlot[Varradius, PlotStyle → Red, PlotRange → Full,
    PlotLegends → {"Změřený poloměr proměnné hvězdy"}]];
```

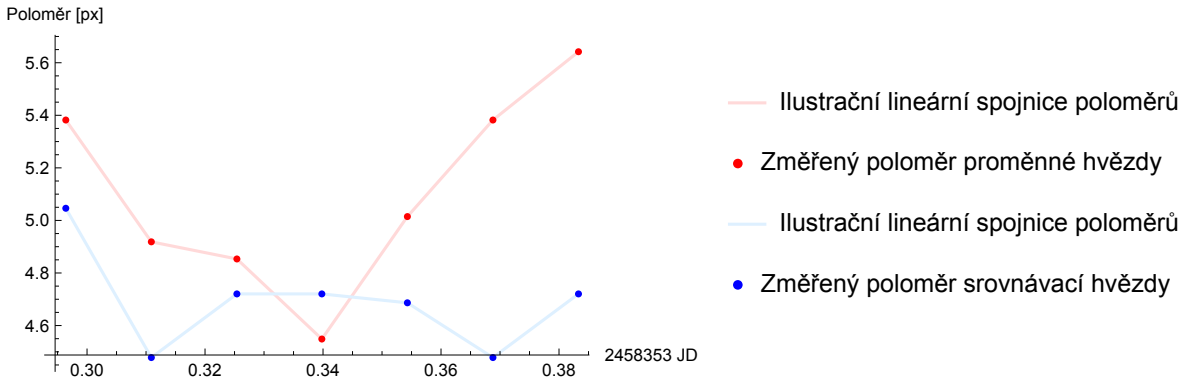
In[176]:=
```
Compradius = {{0.2964, 5.046265044040321},
    {0.3109, 4.478115991081385}, {0.3254, 4.720348719413148},
    {0.3398, 4.720348719413148}, {0.3543, 4.686510657907603},
    {0.3688, 4.478115991081385}, {0.3833, 4.720348719413148}};
```

In[177]:=
```
Compgraf = Show[ListPlot[Compradius, Joined → True,
     PlotLegends → {"Ilustrační lineární spojnice poloměrů"},
     PlotStyle → LightBlue, AxesLabel → {"2458353 JD", "Poloměr [px]"}],
    ListPlot[Compradius, PlotStyle → Blue, PlotRange → Full,
     PlotLegends → {"Změřený poloměr srovnávací hvězdy"}]];
```

Pomocí **Show** provedeme srovnání obou dvou hvězd.

In[178]:=
```
Show[Vargraf, Compgraf]
```

Out[178]=



Na výsledném grafu pozorujeme, že se nám mění poloměr srovnávací hvězdy. Změna poloměru může být způsobena turbulencemi atmosféry (seeing) nebo tzv. bloomingem. Musíme tedy počítat s těmito možnými vlivy na změnu poloměru. U proměnné hvězdy vidíme, jak se mění poloměr v čase, čili jak se změnou hvězdné velikosti při primárním minimu klesá a roste poloměr hvězdy na daném snímku.

4.1.4 Světelná křivka hvězd

Nejprve budeme vycházet z hodnot pro vybrané snímky v předchozí kapitole 4.1.2, kde hodnoty magnitud byly převzaty z programu *Muniwin*. V prvním kroku sestavíme graf magnitudy v čase. “Varmag” jsou hodnoty pro proměnnou hvězdu, “Vargraf” následně graf pro proměnnou hvězdu, “Compmag” hodnoty pro srovnávací hvězdu a graf pojmenovaný jako “Compmaggraf” pro srovnávací hvězdu.

In[179]:=
```
Varmag = {{0.2964, 13.26}, {0.3109, 13.35}, {0.3254, 13.61},
    {0.3398, 13.91}, {0.3543, 13.72}, {0.3688, 13.50}, {0.3833, 13.40}};
```

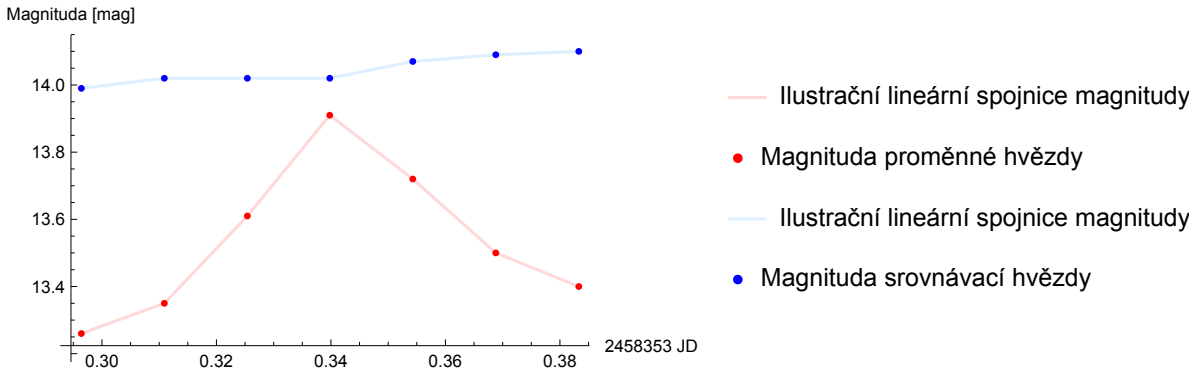
In[180]:=
```
Varmaggraf = Show[ListPlot[Varmag, Joined → True,
     PlotLegends → {"Ilustrační lineární spojnice magnitudy"},
     PlotStyle → LightRed, AxesLabel → {"2458353 JD", "Magnituda [mag]"}],
    ListPlot[Varmag, PlotStyle → Red, PlotRange → Full,
     PlotLegends → {"Magnituda proměnné hvězdy"}]];
```

In[181]:=
```
Compmag = {{0.2964, 13.99}, {0.3109, 14.02}, {0.3254, 14.02},
    {0.3398, 14.02}, {0.3543, 14.07}, {0.3688, 14.09}, {0.3833, 14.10}};
```

In[182]:=
```
Compmaggraf = Show[ListPlot[Compmag, Joined → True,
    PlotLegends → {"Ilustrační lineární spojnice magnitudy"},
    PlotStyle → LightBlue, AxesLabel → {"2458353 JD", "Magnituda [mag]"}],
   ListPlot[Compmag, PlotStyle → Blue, PlotRange → Full,
    PlotLegends → {"Magnituda srovnávací hvězdy"}]];
```

In[183]:=
```
Show[Varmaggraf, Compmaggraf, PlotRange -> Full]
```

Out[183]=



Na výsledném grafu můžeme pozorovat změnu magnitudy v čase proměnné hvězdy, tedy primární minimum. Vzhledem k velké nepřesnosti křivky bude v dalším kroku sestavem graf ze všech naměřených hodnot proměnné a srovnávací hvězdy.

Neprve vložíme získaná data z programu *Muniwin,* která popisují rozdíl změny magnitudy proměnné a srovnávací hvězdy v čase a pojmenujeme je “Data”. Následně hodnoty budeme fitovat. Světelnou křivku lze fitovat pomocí polynomiální funkce

$$y = f(t) = \beta_0 + \beta_1 t + \beta_2 t^2 + \beta_3 t^3 + \ldots + \beta_n t^n, \tag{4}$$

kde t označuje čas v Juliánském datu a β_n je koeficient [31]. Fitovat budeme polynomem 10. stupně. Lze říci, že čím vyšší číslo polynomu, tím přesněji fit popisuje naměřené hodnoty. Nejprve spočítáme nelineární model fitu “Nlmdata” pomocí funkce **NonlinearModelFit** a uvedeme tabulku parametrů pro daný model fitu pomocí **ParameterTable**. Jako poslední uvedeme přesnost fitu “Presnost” využitím **RSquared**.

In[184]:=
```
Data = Import["VCdata.dat"];
```

In[185]:=
```
Nlmdata =
 NonlinearModelFit[Data, {a + b * x + c * x^2 + d * x^3 + e * x^4 + f * x^5 + g * x^6 +
    h * x^7 + i * x^8 + j * x^9 + k * x^10}, {a, b, c, d, e, f, g, h, i, j, k}, x]
```

Out[185]=
```
FittedModel[ -1.59871 × 10^7 + ≪13≫ + 1.03564 × 10^12 x^10 ]
```

In[186]:=
```
Nlmdata["ParameterTable"]
```

Out[186]=

	Estimate	Standard Error	t-Statistic	P-Value
a	-1.59871×10^{7}	508 823.	−31.4197	5.11421×10^{-41}
b	3.99687×10^{8}	1.13774×10^{7}	35.13	5.37985×10^{-44}
c	-4.23514×10^{9}	1.08521×10^{8}	−39.0258	7.84375×10^{-47}
d	2.39895×10^{10}	5.68137×10^{8}	42.2248	5.67199×10^{-49}
e	-7.11154×10^{10}	1.72903×10^{9}	−41.1303	2.94291×10^{-48}
f	4.93671×10^{10}	2.86984×10^{9}	17.202	6.91452×10^{-26}
g	4.13117×10^{11}	1.65911×10^{9}	248.999	1.39996×10^{-98}
h	-1.65608×10^{12}	1.78242×10^{9}	−929.119	9.72298×10^{-136}
i	2.95387×10^{12}	1.92922×10^{9}	1531.12	7.71749×10^{-150}
j	-2.70766×10^{12}	2.14973×10^{9}	−1259.53	2.50787×10^{-144}
k	1.03564×10^{12}	2.64156×10^{9}	392.056	2.18779×10^{-111}

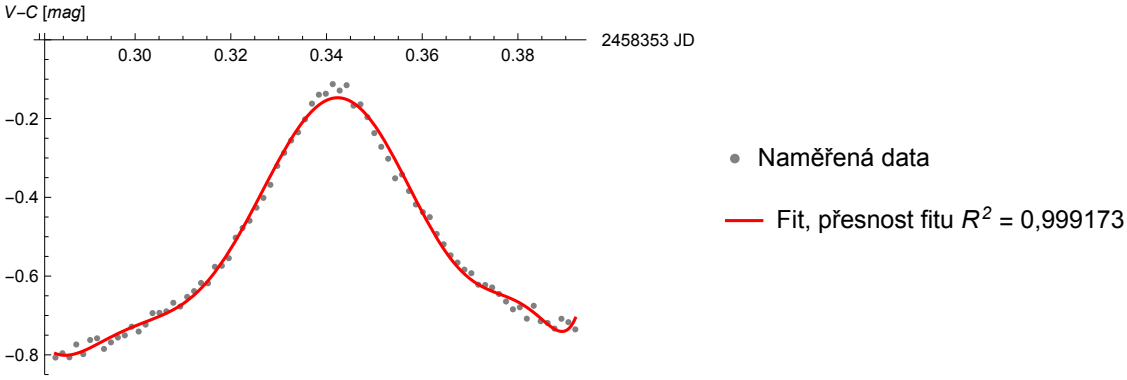
In[187]:=
```
Presnost = Nlmdata["RSquared"]
```

Out[187]=
```
0.999173
```

Nyní již lze sestavit graf "VC" ze získaných hodnot včetně fitu.

In[188]:=
```
VC = Show[ListPlot[Data, PlotLegends → {"Naměřená data"}, PlotStyle → Gray],
  Plot[Nlmdata[x], {x, 0.283362`, 0.3919975}, PlotStyle → Red,
   PlotLegends → {"Fit, přesnost fitu R² = 0,999173"}],
  AxesLabel → {"2458353 JD", "V-C [mag]"}]
```

Out[188]=



Na grafu je znázorněn časový průběh rozdílu magnitud proměnné a porovnávací hvězdy, primární minimum. Na grafu můžeme pozorovat, že primární minimum nastane okolo 2458353,34 JD.

4.2 Analýza fotografie

V této kapitole se budeme zabývat analýzou fotografie, provedeme analýzu objektů na fotografii a ve druhé části analýzu osob na zvolené fotografii astronautů, která je převzata z [32].

4.2.1 Analýza objektů na fotografii

Nejprve vybranou fotografii "Foto" importujeme pomocí **Import** a provedeme náhled pomocí **Thumbnail**. Dále můžeme zobrazit histogram barev a dominantní barvy na fotografii.

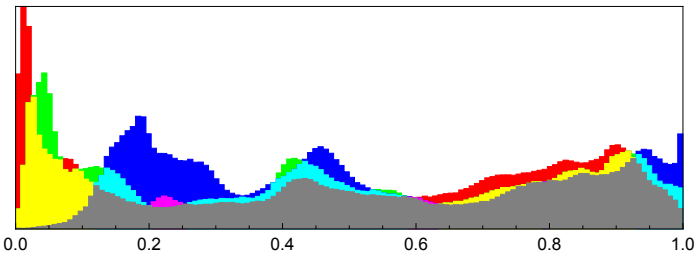
In[189]:=
```
Foto = Import["Foto.jpg"];
```

In[190]:=
```
Thumbnail[Foto]
```

Out[190]=

In[191]:=
```
ImageHistogram[Foto]
```

Out[191]=



In[192]:=
```
DominantColors[Foto]
```

Out[192]= {, , , , , , }

Funkci **ImageIdentify** využujeme k identifikaci objektů na fotografii. K rozpoznání objektů na fotografii lze pomocí funkce **ImageCases**. Mathematica od verze 12 umožňuje nalezení konkrétního objektu v obrázku.

In[193]:=
```
ImageIdentify[Foto, All, 5, "Probability"]
```

Out[193]=
```
⟨| spacesuit → 0.956284, pressure suit → 0.956284,
 protective garment → 0.956315, clothing → 0.956332, consumer goods → 0.956333 |⟩
```

Na výstupu funkce **ImageIdentify** vidíme 5 rozpoznaných objektů a s jakou pravděpodobností byly rozeznány.

Pro přehled vlastností identifikovaných objektů použijeme funkci **ImageContents**.

In[194]:=
```
ImageContents[Foto]
```

Out[194]=

Image	Concept	BoundingBox	Pr
	person	Rectangle[{3.07576, 0.369765}, {496.469, 783.205}]	0.
	person	Rectangle[{688.996, 13.5893}, {1184.46, 692.487}]	0.
	person	Rectangle[{333.429, 33.169}, {852.337, 920.89}]	0.

Dle výstupu vidíme, že na fotografii byli nalezeny 3 osoby. Dále je zde ohraničení daných osob a jejich pravděpodobnost detekovaného objektu.

4.2.2 Analýza osob na fotografii

Pomocí funkcí ve Wolfram Mathematice můžeme provést analýzu osob na fotografii. Pomocí funkce **FindFaces** dokážeme detekovat obličeje. **BoundingBoxArea** v zápisu funkce dám udává velikost plochy na obrázku, kterou zabírá daný obličej. Výstupem dostaneme seznam obsahující detekované obličeje a jejich plochu na obrázku.

In[195]:=
```
FindFaces[Foto, {"Image", "BoundingBoxArea"}] // Dataset
```

Out[195]=

Image	BoundingBoxArea
	15582.0
	17427.0
	18252.0

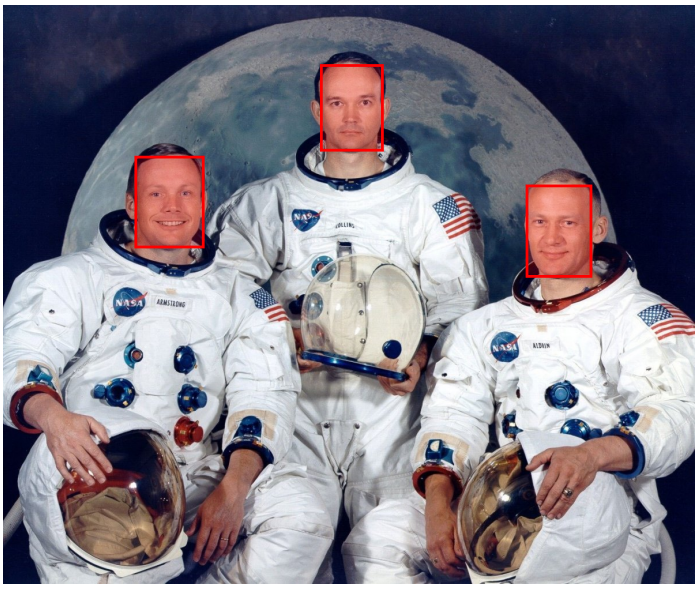
Na dané fotografii lze též zvýraznit obličeje. Nejprve pomocí funkce **FindFaces** provedeme detekci, zobrazí se nám na výstupu v seznamu hodnoty jednotlivých pozic obdélníku, v nichž je detekovaný obličej. Následně jejich hodnoty vložíme do funkce **HiglightImage** a ta zvýrazní dané obličeje na fotografii.

In[196]:=
```
Faces = FindFaces[Foto]
```

Out[196]=
```
{Rectangle[{551.5, 752.5}, {657.5, 899.5}],
 Rectangle[{908.5, 533.5}, {1019.5, 690.5}],
 Rectangle[{229.5, 584.5}, {346.5, 740.5}]}
```

In[197]:=
```
HighlightImage[Foto, Faces, ImageSize → Medium]
```

Out[197]=



Další informace ohledně detekových osob poskytne funkce **FacialFeatures**, a to věk osoby, pohlaví a její emoci na fotografii. Dále tato funkce umožňuje na fotografii zvýraznit pomocí bodů například obrys obličeje (“**OutlinePoints**”), nos (“**NosePoints**”), obočí (“**LeftEyebrowPoints**”, “**RightEyebrowPoints**”), oko (“**LeftEyePoints**”, “**RightEyePoints**”) nebo body vnitřní či vnější části úst (“**MouthInternalPoints**”, “**MouthExternalPoints**”). Na zvolené fotografii provedeme zvýraznění obrysu obličeje.

In[198]:=
```
FacialFeatures[Foto] // Dataset
```

Out[198]=

Image	Age	Gender	Emotion
	37	Male	neutral
	45	Male	neutral
	46	Male	happiness

In[199]:=
```
FacialFeatures[Foto, "OutlinePoints"];
```

In[200]:=
```
HighlightImage[Foto, {PointSize[0.01], %}, ImageSize → Medium]
```

Out[200]=

Mathematica dále umožňuje pomocí "**NotablePerson**" určit osobu na fotografii, ovšem musí se jednat o známou osobu v USA. Dále "**TopProbabilities**" určí zvolený počet nejpravděpodobnějších osob a jejich pravděpodobnost.

In[201]:=
```
Person = ;
```


In[202]:=
```
Classify["NotablePerson", Person, {"TopProbabilities", 5}] // Dataset
```

Out[202]=

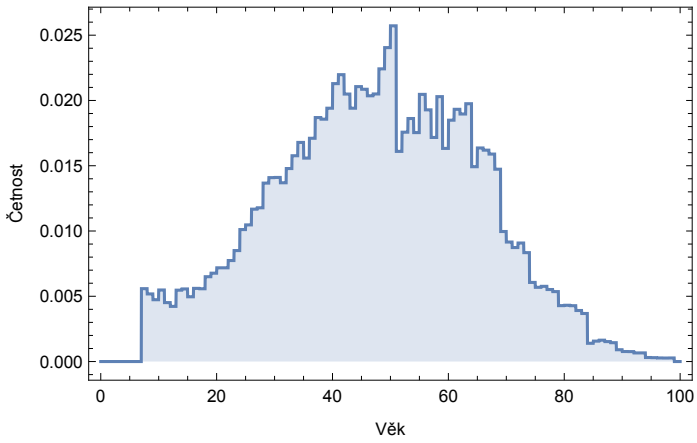
Neil Armstrong	→ 0.879995
Neil Patrick Harris	→ 0.0953584
Fedor Emelianenko	→ 0.00696456
Glen Jacobs	→ 0.00313364
Condoleezza Rice	→ 0.00286691

Podle výstupu se na daném výřezu fotografie s pravděpodobností 0,879995 nachází Neil Amstrong.

Jestliže známe křestní jméno osoby, lze pomocí **Predict** a argumentu "**NameAge**" zobrazit nejpravděpodobnější věk jména osoby. Pomocí "**Distribution**" lze získat histogram rozdělení věku pro dané jméno a za pomocí **Plot** můžeme zobrazit graf tohoto histogramu.

```
In[203]:= Predict["NameAge", "Neil"]
Out[203]= 51

In[204]:= Distribuce = Predict["NameAge", "Neil", "Distribution"]
Out[204]= DataDistribution[ Type: Histogram
                            Data points: 94 ]

In[205]:= Plot[PDF[Distribuce, x], {x, 0, 100}, Exclusions → None, Filling → Bottom,
           PlotRange → All, Frame → True, FrameLabel → {"Věk", "Četnost"}]
```

Out[205]=



Vidíme, že nejpravděpodobnější věk pro jméno Neil je 51 let. Na výsledném grafu četnosti daného jména na věku vidíme, že nejvíce osob se jménem Neil má věkové rozpětí 40-60 let.

Dále můžeme zrekonstruovat 3D model obličeje. Rekonstrukce je založena na využití regresivního modelu Wolfram Neural Net Repository pomocí principu 3D rekonstrukce popsaného na Wolfram Language, Machine Learning for Images - Facial 3D Reconstruction [33].

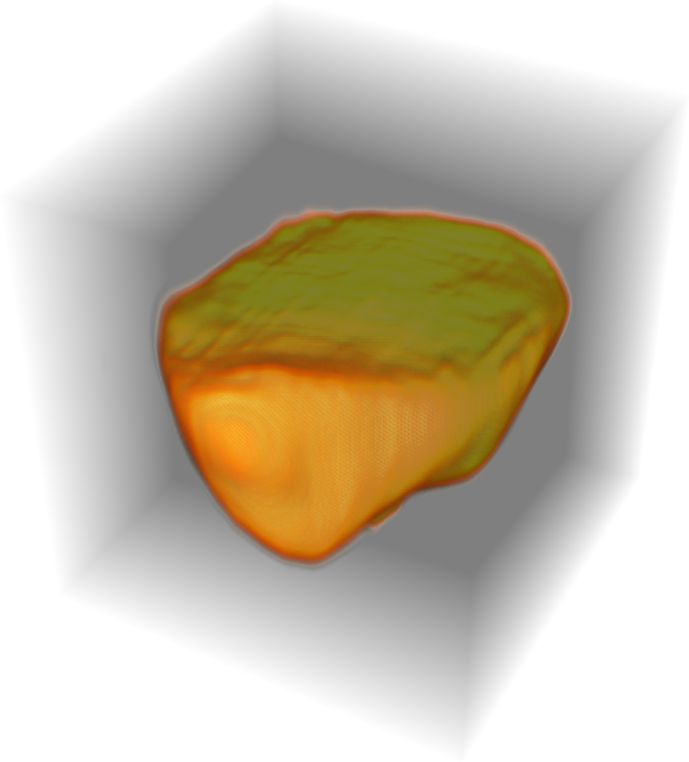
Nejprve použijeme příkaz **NetModel**, který z Wolfram Neural Net Repository stáhne data pro vytvoření 3D rekonstrukce obličeje. "Face3D" pomocí Image3D sestaví 3D výstup obličeje z daného "Netmodel". V dalším kroku "Reconstruction" změníme velikost 3D obrazu, aby jeho rozměry výšky a šířky odpovídaly rozměrům ve 2D.

```
In[206]:= Netmodel =
           NetModel["Unguided Volumetric Regression Net for 3D Face Reconstruction"]
Out[206]= NetChain[ Input port:              image
                    Output port:             array (size: 200 × 192 × 192)
                    Number of layers:        10 ]

In[207]:= Face =
```

```
;
```

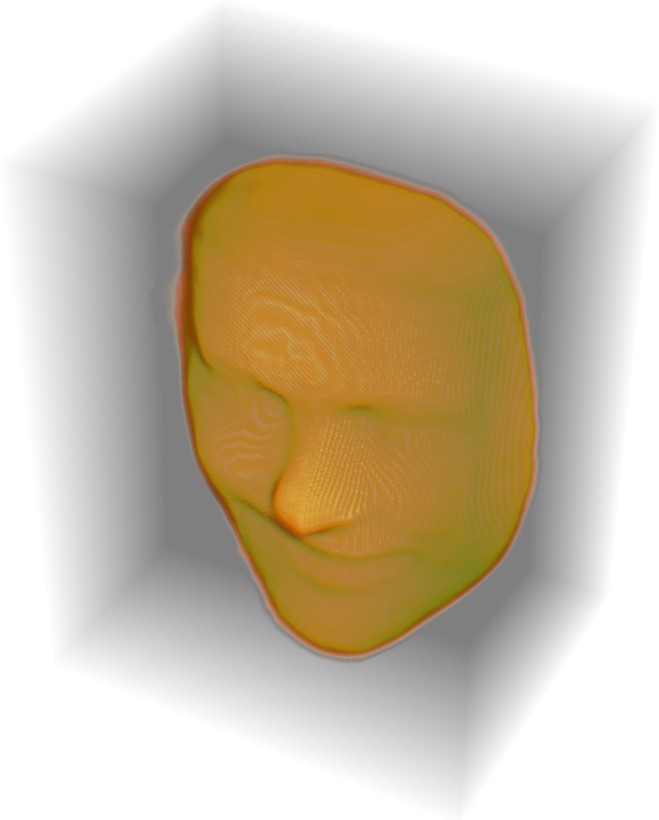
In[208]:=
```
Face3D = Image3D[Netmodel[Face]]
```

Out[208]=



In[209]:=
```
Reconstruction = ImageReflect[
  ImageResize[Face3D, Append[ImageDimensions[Face], 110]], Bottom → Front]
```

Out[209]=



V “Color” dochází k vynásobení původního výřezu obličeje barevným řezem. Vybrané řezy jsou výstupem “ColorPart”.

In[210]:=
```
Color = Map[SetAlphaChannel[Face, #] &, Image3DSlices[Reconstruction, All, 2]];
```

In[211]:=
```
ColorPart = Part[Color, {40, 50, 60}]
```

Out[211]=

V posledním kroku “Face3DColor” dochází k sestavení 3D obrázku z daných řezů.

In[212]:=
```
Face3DColor = ImageReflect[Image3D[Color, ViewPoint → {1, -2, 0},
    ViewAngle → 20 °, ImageSize → 200], Bottom → Front]
```

Out[212]=

Výsledný zrekonstruovaný 3D obličej lze z různých úhlů otáčet a prohlížet.

Závěr

Tato bakalářská práce se zabývá analýzou obrazu a jejího využití v praxi v progamu Wolfram Mathematica. První část práce je zaměřena na historii a strukturu programu Wolfram Mathematica a na principy programování Wolfram Language.

V druhé části (analýze obrazu) práce seznamuje s nejběžnějšími datovými formáty a jejich vlastnostmi, barevnými vlastnostmi obrazu a s charakteristikami obrazu, mezi které je zařazena hloubka obrazu, dynamický rozsah, jas, kontrast a histogram.

Třetí část práce uvádí přehled funkcí pro zpracování 2D obrazu. Neprve se zaměřuje na vytvoření a import obrazu do Notebooku. Dále jsou zde funkce rozděleny do několika kapitol, popisující vlastnosti a zobrazení obrazu, základní manipulaci s obrazem, barevné zpracování obrazu, geometrické operace, morfologické zpracování obrazu, filtry, funkce pro detekci objektů a bodů zájmu v obraze a též počítačové vidění, které se zaměřuje na detekci a rozpoznávání objektů.

Čtvrtá část práce je zaměřena na využití funkcí v praxi, které byly použity na zkoumání proměnné hvězdy V1207 Her. Nejprve byl původní snímek zpracován pro nalezení nejjasnějších hvězd. Dále bylo provedeno měření poloměru proměnné a srovnávací hvězdy. U srovnávací hvězdy byla zjištěna změna poloměru v čase, ačkoliv byl očekáván neproměnný poloměr. Tato změna nejpravděpodobněji je způsobena turbulencemi v atmosféře (tzv. seeing) a vlastnosti CCD čipu, tzv. blooming, kdy dochází k “přelévání” elektronů do sousední nábojové studně, což mohlo vést k naměření větší hodnoty poloměru. U proměnné hvězdy V1207 Her bylo pozorováno, že se změnou magnitudy se mění poloměr. Čím byla hvězda slabší, tím se zmenšoval na snímku její poloměr. V závěrečné části byla zpracována naměřená data, rozdíl magnitud proměnné a srovnávací hvězdy v čase, k pozorování primárního minima na výsledném grafu.

Další použití funkcí bylo provedeno na analýze fotografie, k detekci objektů a osob na fotografii. Byl detekován obličej, následné zjištění jména dané osoby a zobrazení pro křestní jméno graf četnosti osob s daným jménem podle věku. Nakonec byl zrekonstruován 3D model obličeje dané osoby.

Seznam použitých zdrojů

[1] Wikipedia: *Stephen Wolfram* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2019-07-14]. Dostupné z: https://en.wikipedia.org/wiki/Stephen_Wolfram

[2] WOLFRAM, Stephen. *Stephen Wolfram|Blog: There Was a Time before Mathematica…* [online]. 2013 [cit. 2019-07-14]. Dostupné z: https://blog.stephenwolfram.com/2013/06/there-was-a-time-before-mathematica/

[3] ABBASI, Nasser M. *A little bit of Mathematica history* [online]. 2019 [cit. 2019-07-13]. Dostupné z: https://www .12000.org/my_notes/compare_mathematica/index.htm

[4] Wikipedia: *Wolfram Alpha* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2019-07-14]. Dostupné z: https://cs.wikipedia.org/wiki/Wolfram_Alpha

[5] BOUŠKA, Martin. *MATHEMATICA – příručka s příklady pro učitele a studenty, METODA 2009* [online]. Praha, 2009 [cit. 2019-07-14]. Dostupné z: https://www.pslib.cz/jaromir.oscadal/PRG%20-V1/Mathematica%20-%20 př % C3 % ADručka3.pdf

[6] Wikipedia: *Wolfram Language* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2019-07-14]. Dostupné z: https://en.wikipedia.org/wiki/Wolfram_Language

[7] RUSKEEPÄÄ, Heikki. *Mathematica® Navigator: Mathematics, Statistics, and Graphics*. 3rd ed. Department of Mathematics University of Turku, Finland: Elsevier, 2009. ISBN 978-0-12-374164-6.

[8] ŠRÁMEK, Jaromír, Ondřej RÁČEK, Martin SEDLÁŘ a Vojtěch MORNSTEIN. *MUNI MED | Lékařská fakulta Masarykovy univerzity: Získávání a analýza obrazové informace* [online]. Brno, 2011 [cit. 2019-07-27]. Dostupné z: https://www.med.muni.cz/biofyz/Image/ucebnice.pdf

[9] University information system MENDELU. *Formáty souborů* [online]. [cit. 2019-07-27]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=22297

[10] BECHYŇOVÁ, Marta. Stránky k výuce informatiky. *Grafické formáty* [online]. 2019 [cit. 2019-06-27]. Dostupné z: http://www.ivt.mzf.cz/grafika/graficke-formaty/

[11] BARON, Prokop. DIGIarena. *Formáty obrázků* [online]. 2006 [cit. 2019-06-27]. Dostupné z: https://digiarena.zive.cz/formaty-obrazku_ 5

[12] Dalekohledy-fomei. *Světlo není jen bílé* [online]. [cit. 2019-07-13]. Dostupné z: https://www.dalekohledy-fomei.cz/clanky-a-novinky/svetlo-neni-jen-bile/

[13] BECHYŇOVÁ, Marta. Stránky k výuce informatiky. *Barevné modely* [online]. 2019 [cit. 2019-07-13]. Dostupné z: http://www.ivt.mzf.cz/grafika/barevne-modely/

[14] HONSNEJMAN, Petr. Moje Tajemno. *Barevná hloubka* [online]. Česká Lípa, 2016 [cit. 2019-07-13]. Dostupné z: http://moje.tajemno.net/barevna-hloubka/

[15] University information system MENDELU. *Pixel* [online]. [cit. 2019-07-14]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=6335

[16] Wikipedia: *Barevná hloubka* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2019-07-13]. Dostupné z: https://cs.wikipedia.org/wiki/Barevná_hloubka

[17] KOVALČÍK, Vít. Milujeme fotografii. *Přepálené fotografie? Pochopte, co je dynamický rozsah, a jsou minulostí* [online]. 2016 [cit. 2019-07-13]. Dostupné z: https://www.milujemefotografii.cz/prepalene-fotografie-pochopte-co-je-dynamicky-rozsah-a-jsou-minulosti

[18] HONSNEJMAN, Petr. Moje Tajemno. *Barevná hloubka* [online]. Česká Lípa, 2016 [cit. 2019-07-13]. Dostupné z: https://moje.tajemno.net/dynamicky-rozsah/

[19] PIHAN, Roman. FotoRoman. *Vše o světle - 3. intenzita (jas) světla* [online]. Praha, 2012 [cit. 2019-07-13]. Dostupné z: http://www.fotoroman.cz/tech2/svetlo03jas.htm

[20] IT Slovník. Kontrast [online]. [cit. 2019-07-13]. Dostupné z: https://it-slovnik.cz/pojem/kontrast

[21] Wikipedia: *Princip kontrastu* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2019-07-13]. Dostupné z: https://cs.wikipedia.org/wiki/Princip_kontrastu

[22] Wolfram Research, Inc. *Wolfram Language & System Documentation Center* [online]. Champaign, Illinois [cit. 2018-09-05]. Dostupné z: https://reference.wolfram.com/language/

[23] MIKULÁŠEK, Zdeněk a Miloslav ZEJDA. *Úvod do studia proměnných hvězd* [online]. Brno: Masarykova Univerzita, 2013 [cit. 2019-07-31]. ISBN 978-80-210-6241-2. Dostupné z: http://physics.muni.cz/~zejda/PHV.pdf

[24] Astromia. *Proměnné hvězdy* [online]. 2010 [cit. 2019-08-02]. Dostupné z: http://hvezdy.astro.cz/promenne/704-promenne-hvezdy

[25] Hvězdárna Valašské Meziříčí. *Proměnné hvězdy* [online]. [cit. 2019-08-13]. Dostupné z: https://www.astrovm.cz/cz/odborna-cinnost/promenne-hvezdy.html

[26] BRÁT, Luboš, Ladislav ŠMELCER a Jaroslav TRNKA. *Proměnné hvězdy a možnosti jejich pozorování a výzkumu* [online]. Valašské Meziříčí, 2011 [cit. 2019-08-02]. Dostupné z: https://www.astrovm.cz/userfiles/file/projekty/kosoap/vzdelavaci_metodicky_material_KOSOAP-PH-maly.pdf

[27] HADERKA, Ondřej. *Přístroje pro astronomii 1: Závěr kursu* [online]. [cit. 2019-07-30]. Dostupné z: http://jointlab.upol.cz/haderka/pa1/Př% C3 % ADstroje %20 pro %20 astronomii %201 %20-%20 konec.ppt

[28] Wikipedia: *Hvězda typu W Ursae Majoris* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2019-08-03]. Dostupné z: https://cs.wikipedia.org/wiki/Hvězda_typu_W _Ursae _Majoris

[29] Sekce pozorovatelů proměnných hvězd České astronomické společnosti: O-C gateway. *Hvězda: V1207 Her* [online]. [cit. 2019-08-01]. Dostupné z: http://var2.astro.cz/ocgate/ocgate.php?star=V1207 %20 Her&lang=cz&cons=Her #ocdiagram

[30] Wikipedia: *FITS* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2019-08-03]. Dostupné z: https://cs.wikipedia.org/wiki/FITS

[31] BENN, David. *VStar version 2.20.0: User Manual* [online]. [cit. 2019-07-31]. Dostupné z: https://www.aavso.org/files/vstar/VStarUserManual.pdf

[32] PŘIBYL, Tomáš. Kosmonautix. *Týden s Jedenáctkou (3) Vznik posádky Apolla 11 aneb kterak Neil Armstrong připravil Jamese Lovella o procházku po Měsíci* [online]. 2019 [cit. 2019-08-03]. Dostupné z: https://www.kosmonautix.cz/2019/07/tyden-s-jedenactkou-3-vznik-posadky-apolla-11-aneb-kterak-neil-armstrong-pripravil-jamese-lovella-o-prochazku-po-mesici/

[33] Wolfram Research, Inc. Wolfram Language. *Facial 3D Reconstruction* [online]. Champaign, Illinois, 2019 [cit. 2019-07-25]. Dostupné z: https://www.wolfram.com/language/12/machine-learning-for-images/facial-3d-reconstruction.html?product=language