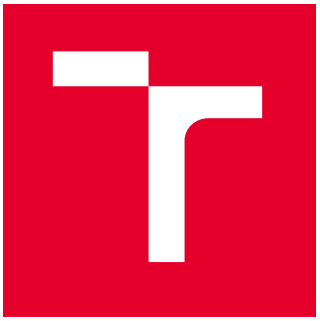
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta strojního inženýrství

DIPLOMOVÁ PRÁCE

Brno, 2023 Bc. Karolína Gebrtová



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY

INSTITUTE OF MATHEMATICS

ROZKLAD VIDEOSEKVENCÍ NA VÍCE SLOŽEK S RŮZNOU DYNAMIKOU

DECOMPOSITION OF VIDEO-SEQUENCES INTO COMPONENTS WITH DIFFERENT DYNAMICS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Karolína Gebrtová**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Mgr. Pavel Rajmic, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav matematiky
Studentka:	**Bc. Karolína Gebrtová**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Matematické inženýrství
Vedoucí práce:	**prof. Mgr. Pavel Rajmic, Ph.D.**
Akademický rok:	2022/23

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Rozklad videosekvencí na více složek s různou dynamikou

Stručná charakteristika problematiky úkolu:

Videosekvenci, ve které se dějí různě dynamické jevy (např. pozadí, pomalý pohyb, rychlý pohyb), lze chápat jako posloupnost součtů n–tic snímků, kdy každá z této n–tice vykazuje odlišný charakter dynamiky. Při vhodném uspořádání pixelů z jednotlivých snímků lze tyto složky identifikovat pomocí různých metod, od jednoduchých po složitější. Cílem práce bude nastudovat problematiku, navrhnout a implementovat různé separační algoritmy (v Matlabu), aplikovat na simulovaná a reálná data a porovnat mezi sebou.

Cíle diplomové práce:

– Naučit se zacházet s videodaty v Matlabu (import, zpracování, export).
– Nastudovat problematiku separace dynamické a statické složky ve videu, nejprve obecně.
– Zaměřit se posléze na jednodušší metody (mediánová, průměrovací), a postupně přejít k metodám a modelům složitějším (metody využívající tzv. řídkých a nízkohodnostních reprezentací, metody založené na různé penalizace různě dynamických komponent, metoda Dynamic Mode Decomposition, AR/ARMA modely a podobně).
– Implementovat tyto separační úlohy v Matlabu, optimalizovat a použít na simulovaná a reálná data.
– Porovnat výstupy objektivními metrikami (kde je to možné) a subjektivně.
– Porovnat výpočetní náročnost metod.
– V případě reálných dat se zaměřit na záznamy kamer ze silničního provozu a na videa sluneční korány prof. Druckmüllera.

ABSTRAKT

Diplomová práce je zaměřena na rozklad videosekvencí – převážně na separaci pozadí a dynamické složky, kde pozadí zůstává stejné a pouze malé části jsou v pohybu. Takové video je reprezentováno nízkohodnostní a řídkou složkou. Díky nízkohodnostní struktuře může být pozadí odseparováno pomocí mediánového filtru a metody dynamických modů. Dále bude využita tzv. robustní analýza hlavních komponent, která je formulována jako optimalizační úloha, a její nekonvexní varianta. Zmíněné je i multiresolution DMD, které je schopno rozkladu na více dynamických složek podle jejich rychlosti.

KLÍČOVÁ SLOVA

videosekvence, rozklad videosekvencí, separace pozadí, statická a dynamická složka, řídké reprezentace, nízkohodnostní struktura, singulární rozklad, mediánový filtr, principal component pursuit, nekonvexní robustní analýza hlavních komponent, robustní analýza hlavních komponent, metoda dynamických modů, multiresolution dynamic mode decomposition

ABSTRACT

The thesis is focused on the decomposition of video sequences, primarily on background and dynamic component separation, where the background remains the same and only small parts are in motion. Such a video is represented by a low-rank and sparse component. Thanks to the low-rank structure, the background can be separated using a median filter and dynamic mode decomposition. Furthermore, the robust principal component analysis, formulated as an optimization problem and its non-convex variant, will be employed. Also mentioned will be the multiresolution DMD, which is capable of decomposing the dynamic component based on its velocity.

KEYWORDS

videosequance, videosequance decomposition, background separation, dynamic and static structures, sparse representations, low-rank structures, singular value decomposition, median filter, principal component pursuit, nonconvex robust principal component analysis, robust principal component analysis, dynamic mode decomposition, multiresolution dynamic mode decomposition

GEBRTOVÁ, Karolína. *Rozklad videosekvencí na více složek s různou dynamikou*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav matematiky, 2023, 109 s. Diplomová práce. Vedoucí práce: prof. Mgr. Pavel Rajmic, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Karolína Gebrtová
VUT ID autora:	200875
Typ práce:	Diplomová práce
Akademický rok:	2022/23
Téma závěrečné práce:	Rozklad videosekvencí na více složek s různou dynamikou

Prohlašuji, že svou závěrečnou práci jsem vypracovala samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autorka uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autorky*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu diplomové práce panu prof. Mgr. Pavlovi Rajmicovi, Ph.D. za odborné vedení, časté konzultace a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Tématem této diplomové práce je rozklad videosekvencí na více složek s různou dynamikou. Nejčastěji užívaným rozkladem je rozklad videosekvence na dvě složky – statické pozadí a dynamická pohybující se složka. Součástí této práce je i rozklad dynamické složky dle rychlosti jednotlivých pohybujících se objektů. V současnosti se téma rozkladu videosekvencí těší širokému množství užití. Aplikace nalezneme především v oblasti monitorovacích videosystémů, protože použitím těchto metod jsme schopni detekovat osoby, vozidla a jiné pohybující se objekty. Další možnost aplikace leží v oblasti rozpoznávání objektů, kde jsme díky separaci schopni z obrazů odstranit i stíny a jiná zkreslení (dynamickou složku) a tím lépe detekovat např. obličeje lidí nebo defekty výrobků. Obdobně lze rozklad videosekvencí využít i jako základ pro sledování objektů (object tracking). Techniky rozkladu jsou užitečné i v jiných odvětvích např. při postprodukci a střihu videí. Při současném trendu, kdy se využití umělé inteligence stává běžnou záležitostí, lze rozklad videosekvencí použít k rozpoznávání gest a k obecné interakci člověka s počítačem.

V této práci se omezíme pouze na videa natočená statickou kamerou. Příkladem může být pevná kamera snímající provoz na silnici. V takovém případě je pozadí téměř neměnné. Vhodným uspořádáním jednotlivých pixelů lze zjistit, že se pozadí chová jako nízkohodnostní struktura, jejíž jedinou změnu působí dynamická složka. Dále budeme předpokládat, že objekty dynamické složky jsou malé a není jich příliš mnoho, dynamická složka je tedy „řídká“. Využitím výše zmíněných vlastností jsme schopni jednotlivé složky efektivně oddělit.

Představíme zde čtyři základní způsoby rozkladu pro videosekvence ve stupních šedi – mediánový filtr, metoda DMD (Dynamic Mode Decomposition), metoda PCP (Principal Component Pursuit) a metoda RPCA (Robust Principal Component Analysis), u které lze volit mezi konvexní a nekonvexní variantou. Dále bude představeno rozšíření metod pro barevné videosekvence. A také bude zmíněna metoda Multiresolution DMD vycházející z metody DMD. Tato metoda bude schopna rozložit i dynamickou složku dle rychlosti jednotlivých pohybujících se objektů.

Mediánový filtr je pro svoji jednoduchost implementace a rychlost výpočtu velmi využívanou metodou, znát ho můžeme například z programu Photoshop. Metoda PCP je již starší metodou, publikována byla v roce 2011. Metoda je založena na minimalizaci vážené kombinace nukleární normy a ℓ_1 normy. Stále vychází množství publikací, které se snaží tuto metodu optimalizovat a dále vylepšit. Jednou z nich je RPCA dle Chartranda, které dovoluje volit místo konvexní ℓ_1 normy i nekonvexní ℓ_p normy, kde $p \in \langle 0, 1)$. Metoda DMD patří k novějším metodám (první publikace pochází z roku 2014) a těší se velkému užití nejen v oblasti zpracování obrazu, ale také v oborech jako je hydromechanika, neurologie nebo ekonomie. DMD má široké

využití, protože aproximuje složité (často neznámé) nelineární systémy na lineární systémy. V současnosti jsou stále publikována rozšíření DMD metody, která se hodí např. potřebujeme-li systém aproximovat na jednoduchý nelineární systém nebo pro optimální řízení. Jedním takovým rozšířením je i Multiresolution DMD, které je vhodné zejména pro systémy s různou časovou dynamikou.

Cílem této práce je nastudovat výše zmíněné metody rozkladu, implementovat a dále optimalizovat tyto úlohy v Matlabu a následně je použít na simulovaná a reálná data. Výsledky získané jednotlivými metodami budou poté porovnány subjektivně a objektivně pomocí vhodných metrik.

Tato diplomová práce je rozdělena do pěti kapitol. V první kapitole jsou uvedeny základní matematické pojmy, které budou používány v dalších kapitolách. V druhé kapitole budou detailně vysvětleny a odvozeny jednotlivé metody separace pro vstupní data ve stupních šedi. Ve třetí kapitole budou metody z druhé kapitoly rozšířeny pro případ, kdy jsou vstupní data barevná. Čtvrtá kapitola je věnovaná implementaci jednotlivých metod v Matlabu a poslední pátá kapitola srovnává jednotlivé metody na simulovaných a reálných datech podle výpočetní náročnosti a přesnosti separace, která je ověřena vhodnými metrikami.

1 Základní matematické pojmy

1.1 Normy vektorů a matic

Definice 1.1. [28] *Nosičem vektoru* $\mathbf{x}$ *rozumíme množinu jeho indexů, ve kterých má vektor nenulové hodnoty. Tuto množinu budeme značit jako* $\mathrm{supp}(\mathbf{x}) = \{i, x_i \neq 0\}$.

Abychom byli schopni porovnávat velikosti jednotlivých vektorů, zavedeme tzv. *normu* vektoru, kterou lze chápat jako velikost daného vektoru.

Definice 1.2. [16] *Nechť* X *je vektorový prostor nad* $\mathbb{C}$ *a nechť zobrazení* $\|\cdot\| : X \to \langle 0, \infty)$ *splňuje* $\forall \mathbf{x},\mathbf{y} \in X$ *a* $\forall \lambda \in \mathbb{C}$ *podmínky:*

1. $\|\mathbf{x}\| = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$
2. $\|\lambda\mathbf{x}\| = |\lambda|\|\mathbf{x}\|$
3. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$

Pak zobrazení $\|\cdot\|$ *nazveme normou na prostoru* X.

Jelikož norma zobrazí každý vektor na nezáporné číslo, jsme s její pomocí schopni porovnávat velikosti daných vektorů.

Užitečnou třídou vektorových norem jsou tzv. *p-normy.*

Definice 1.3. [16] *Nechť* $p \in \langle 1, \infty)$, *pak jako* ℓ_p *normu vektoru* $\mathbf{x} \in \mathbb{C}^n$ *chápeme číslo*

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}. \tag{1.1}$$

Speciálně pro případ $p = \infty$ je ℓ_∞ definována jako

$$\|\mathbf{x}\|_\infty = \max_i |x_i|. \tag{1.2}$$

Velmi často užívanou normou je ℓ_2 norma, která se také nazývá *Eukleidovská*, protože odpovídá vzdálenosti dvou bodů v Eukleidovském prostoru.

ℓ_p normy jsou velmi využívány v optimalizaci a statistice, protože se jedná o konvexní funkce – často tedy vedou na výpočetně efektivní algoritmy. V této práci se budeme zabývat i použitím nekonvexních algoritmů. Z tohoto důvodu rozšíříme definici ℓ_p normy i pro případ $p \in \langle 0, 1)$.

Definice 1.4. [28] *Pro* $p \in (0, 1)$ *dodefinujeme* ℓ_p *normu vektoru* $\mathbf{x} \in \mathbb{C}^n$ *číslem*

$$\|\mathbf{x}\|_p = \sum_{i=1}^{n} |x_i|^p. \tag{1.3}$$

Speciálně ℓ_0 *normou vektoru* $\mathbf{x} \in \mathbb{C}^n$ *rozumíme číslo udávající počet nenulových složek daného vektoru:*

$$\|\mathbf{x}\|_0 = |\mathrm{supp}(x)|. \tag{1.4}$$

Správně bychom funkci z definice 1.4 neměli nazývat normou, protože nesplňuje předpoklad $\|\lambda\mathbf{x}\| = |\lambda|\|\mathbf{x}\|$. Jedná se tedy spíše o pseudonormu. Přesto k této funkci bude i nadále referováno jako k normě.

V dalších kapitolách se zaměříme především na práci s maticemi, proto je potřeba pojem vektorové normy rozšířit i pro matice [16]. Stejně jako v případě vektorů je norma matic funkce zobrazující matici na nezáporné číslo. Toto zobrazení opět musí splňovat podmínky z definice 1.2. V některé literatuře bývá přidána ještě čtvrtá podmínka $\|\mathbf{AB}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$, ovšem tato podmínka přímo vyplývá z předešlých podmínek.

Poznámka 1.5. *Pokud chceme být schopni použít všechny vektorové normy i pro matice, stačí umět danou matici* $\mathbf{X}$ *převést na vektor (ozn.* vec $\mathbf{X}$*). Matici lze vektorizovat například tak, že jednotlivé sloupce matice „poskládáme pod sebe“ a získáme jeden dlouhý sloupcový vektor. Norma matice potom lze vypočítat jako* $\|\mathbf{X}\| = \|\text{vec } \mathbf{X}\|$.

Jednou z nejčastěji užívaných maticových norem je tzv. *Frobeniova norma* [22], která odpovídá ℓ_2 normě pro matice

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|x_{ij}|^2}. \tag{1.5}$$

V dalších kapitolách budeme kromě vektorových norem potřebovat i tzv. *nukleární normu*.

Definice 1.6. [22] *Nukleární normu* $\|\cdot\|_*$ *matice* $\mathbf{X} \in \mathbb{C}^{m\times n}$ *definujeme jako součet singulárních čísel této matice. Tento součet můžeme také vyjádřit jako* ℓ_1 *normu singulárních čísel.*

$$\|\mathbf{X}\|_* = \|\sigma(\mathbf{X})\|_1 = \sum_{i=1}^{k}\sigma_i, \tag{1.6}$$

kde $k = \min(m, n)$.

Singulární čísla a jejich výpočet bude rozebrán v podkapitole 1.4.

1.2 Řídkost

Definice 1.7. [28] *Vektor* $\mathbf{x} \in \mathbb{C}^n$ *pro* $k \in \mathbb{N}$ *nazveme* k*-řídkým, platí-li*

$$\|\mathbf{x}\|_0 \leq k.$$

Tedy k-řídký vektor má maximálně k nenulových složek. Analogicky lze definovat k-řídkou matici.

Pro účely této práce budeme předpokládat, že nenulové prvky řídké matice jsou rovnoměrně rozloženy napříč celou maticí. Tento předpoklad by měl zajistit, že daná řídká matice nebude mít většinu sloupců nebo řádků nulových. Tzn. matice nebude mít nízkou hodnost.

1.3 Vlastní čísla a vlastní vektory

Definice 1.8. [33] *Nechť* $\mathbf{A}_{n,n}$ *je čtvercová komplexní matice. Vlastní vektor matice* $\mathbf{A}$ *je nenulový vektor* $\mathbf{v} \in \mathbb{C}^n$ *takový, že pro číslo* $\lambda \in \mathbb{C}$ *platí*

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}. \tag{1.7}$$

Číslo λ *se nazývá vlastní číslo matice* $\mathbf{A}$ *a říkáme, že vektor* $\mathbf{v}$ *je vlastní vektor příslušný vlastnímu číslu* λ*.*

Vlastní vektor $\mathbf{v}$ nesmí být nulový, ale vlastní číslo λ může být nulové. Následující věta poslouží jako návod, jak vypočítat vlastní čísla.

Tvrzení 1.9. [16] *Vlastní čísla matice* $\mathbf{A} \in \mathbb{C}^{n\times n}$ *jsou kořeny tzv. charakteristického polynomu*

$$\det(\mathbf{A} - \lambda\mathbf{E}) = 0, \tag{1.8}$$

kde $\mathbf{E}$ *je jednotková matice řádu* n*.*

Důkaz. Rovnice (1.7) může být upravena na tvar $(\mathbf{A} - \lambda\mathbf{E})\mathbf{v} = 0$. Jelikož vlastní vektor $\mathbf{v}$ musí být nenulový a matice $\mathbf{A} - \lambda\mathbf{E}$ je čtvercová, tak jediné netriviální řešení rovnice $(\mathbf{A} - \lambda\mathbf{E})\mathbf{v} = 0$ získáme, když bude matice $\mathbf{A} - \lambda\mathbf{E}$ singulární. Tzn. když determinant $\mathbf{A} - \lambda\mathbf{E}$ bude nulový. □

Pro polynomy s komplexními koeficienty a kořeny platí tzv. *základní věta algebra* [33]. Důsledkem této věty je, že polynom s komplexními koeficienty stupně $n \geq 1$ má v komplexní rovině právě n kořenů (včetně násobnosti daných kořenů). Tedy z věty 1.9 vyplývá, že komplexní matice $\mathbf{A} \in \mathbb{C}^{n\times n}$ má právě n vlastních čísel včetně násobnosti, která jsou obecně komplexní.

1.4 Singulární (SVD) rozklad

Tato podkapitola vychází převážně z knih [16],[35].

Definice 1.10. *Nechť* $\mathbf{A} \in \mathbb{C}^{m\times n}$ *je obecná matice. Singulárním (SVD) rozkladem matice* $\mathbf{A}$ *rozumíme*

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \tag{1.9}$$

kde $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ *je diagonální matice, jejíž diagonální prvky jsou tzv. singulární čísla* σ_i*. Tato čísla jsou nezáporná a seřazena podle velikosti od největšího. Tedy* $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k \geq 0$*, kde* $k = \min(m, n)$*. A matice* $\mathbf{U} \in \mathbb{C}^{m\times m}$*,* $\mathbf{V} \in \mathbb{C}^{n\times n}$ *jsou unitární matice tzv. levých a pravých singulárních vektorů. A* $\mathbf{V}^*$ *je hermitovská transpozice matice* $\mathbf{V}$*.*

Užitečnou vlastností unitárních matic je, že všechny jejich sloupce (řádky) $\mathbf{u}_i$ a $\mathbf{v}_i$ jsou ortonormální vektory (navzájem kolmé s velikostí jedna). Další užitečnou vlastností je, že hermitovská transpozice unitární matice $\mathbf{U}^*$ je rovna její inverzi $\mathbf{U}^{-1}$. Tedy platí $\mathbf{U}\mathbf{U}^* = \mathbf{U}^*\mathbf{U} = \mathbf{E}$, kde $\mathbf{E}$ je jednotková matice.

Nespornou výhodou SVD rozkladu je, že existuje pro všechny matice. Tuto vlastnost zaručuje následující tvrzení.

Tvrzení 1.11. [35] *Pro každou matici $\mathbf{A} \in \mathbb{C}^{m\times n}$ existuje její singulární rozklad. Navíc všechna singulární čísla σ_i matice $\mathbf{A}$ jsou jednoznačně daná (pokud předpokládáme, že jsou vždy seřazena podle velikosti od největšího) a pokud $\mathbf{A}$ je čtvercová a jednotlivá σ_i jsou od sebe navzájem různá, pak levé a pravé singulární vektory $\mathbf{u}_i$ a $\mathbf{v}_i$ jsou jednoznačné až na násobení komplexním číslem velikosti jedna.*

V této práci se budeme zabývat minimalizací hodnosti matic. K tomu bude využit právě SVD rozklad, protože, jak ukazuje následující tvrzení, pomocí SVD rozkladu můžeme snadno zjistit hodnost matice.

Tvrzení 1.12. [16] *Hodnost matice $\mathbf{A}$ je rovna číslu r – počtu nenulových singulárních čísel této matice. Tedy*

$$\text{rank } \mathbf{A} = \|\sigma(\mathbf{A})\|_0 = r, \tag{1.10}$$

kde $\sigma(\mathbf{A})$ je vektor singulárních čísel.

Důkaz tvrzení 1.12 vychází z faktu, že hodnost diagonální matice je určena počtem nenulových prvků této diagonální matice. A jelikož je matice singulárních čísel $\mathbf{\Sigma}$ diagonální, počet nenulových singulárních čísel odpovídá její hodnosti.

Poznámka 1.13. *V praxi se často místo klasického SVD rozkladu používá tzv.* redukovaný SVD rozklad. *Tento rozklad je vhodný, zejména máme-li matici s nízkou hodností r. V tomto případě bude matice $\mathbf{\Sigma}$ mít pouze r nenulových sloupců a řádků a zbytek bude nulový. Při použití redukovaného SVD rozkladu jsou všechny tyto nulové sloupce a řádky $\mathbf{\Sigma}$ ignorovány. Stejně tak se nedopočítávají všechny sloupce a řádky matic $\mathbf{U}, \mathbf{V}$, které by byly danými nulovými sloupci a řádky $\mathbf{\Sigma}$ násobeny. Tedy použitím redukovaného SVD rozkladu namísto klasického stále získáme přesnou reprezentaci původní matice, ale značně snížíme výpočetní náročnost.*

1.5 Pseudoinverzní matice

Pseudoinverzní matice představuje zobecnění inverzní matice pro obecné obdélníkové matice $\mathbf{A} \in \mathbb{C}^{m\times n}$. Pseudoinverzní matice se používá i pro čtvercové singulární matice, protože nejsme schopni spočítat jejich inverzní matici.

Existují dva různé způsoby, jak definovat *Moore–Penroseovu pseudoinverzní matici*. V této práci budeme pro výpočty využívat definici používající SVD rozklad.

Definice 1.14. [44] *Nechť matice* $\mathbf{A} \in \mathbb{C}^{m\times n}$ *a nechť je její singulární rozklad* $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^*$*, pak matici pseudoinverzní k matici* $\mathbf{A}$ *označíme* $\mathbf{A}^+ \in \mathbb{C}^{n\times m}$ *a definujeme předpisem*

$$\mathbf{A}^+ = \mathbf{V}\Sigma^{-1}\mathbf{U}^*, \tag{1.11}$$

kde $\Sigma^{-1} = \widehat{\Sigma}^*$ *a* $\widehat{\Sigma}$ *je diagonální obdélníková matice, jejíž diagonála je rovna vektoru* $[\sigma_1^{-1}, \sigma_2^{-1}, \cdots, \sigma_n^{-1}]$.

Pokud je matice $\mathbf{A}$ čtvercová a regulární, pak existuje matice $\mathbf{A}^{-1}$ inverzní k $\mathbf{A}$ a je stejná jako matice $\mathbf{A}^+$ pseudoinverzní k $\mathbf{A}$. Tedy $\mathbf{A}^{-1} = \mathbf{A}^+$.

1.6 ℓ_1 relaxace

Tato podkapitola vychází z [28] a bude zde prezentována jednoduchá úloha, jejíž myšlenka řešení bude použita v podkapitole 2.3.2 k odvození metody separace. Při tomto odvození je úloha složitější (kromě ℓ_0 normy se zde minimalizuje i nukleární norma). Přesto se dá zjednodušit stejným principem.

Chceme vyřešit soustavu lineárních rovnic $\mathbf{Ax} = \mathbf{y}$ tak, aby vektor řešení $\mathbf{x}$ byl co nejřidší. Danou úlohu lze zapsat následovně

$$\arg\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{vzhledem k} \quad \mathbf{Ax} = \mathbf{y}. \tag{1.12}$$

Jak již bylo řečeno, ℓ_0 norma není konvexní funkce. K řešení této úlohy tedy nelze použít žádnou z řady efektivních metod a algoritmů konvexní optimalizace. Úlohu tedy neumíme vyřešit v rozumném čase. Bude tudíž potřeba i za cenu ztráty přesnosti tuto úlohu aproximovat na konvexní.

Jako možná nejbližší konvexní norma se nabízí ℓ_1 norma (pro $p \geq 1$ jsou ℓ_p normy konvexní). Otázkou tedy je, zda-li je možné úlohu (1.12) převést na úlohu

$$\arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{vzhledem k} \quad \mathbf{Ax} = \mathbf{y}. \tag{1.13}$$

Mnoho prací se zabývá podmínkami, kdy jsou úlohy (1.12) a (1.13) ekvivalentní. Podmínkami ekvivalence jsou určité vlastnosti matice soustavy $\mathbf{A}$. Příkladem takové podmínky je následující tvrzení.

Definice 1.15. [28] *Vzájemná koherence matice* $\mathbf{A}$ *je definována jako největší absolutní normalizovaný skalární součin dvou různých sloupců matice*

$$\mu(\mathbf{A}) = \max_{1\leq i,j\leq N, i\neq j} \frac{|a_i^T a_j|}{\|a_i\|_2 \|a_j\|_2}, \tag{1.14}$$

kde a_i *označuje* i*-tý sloupec matice* $\mathbf{A}$ *a* N *je počet jejích sloupců.*

Tvrzení 1.16. [28] *Pokud má soustava* $\mathbf{Ax} = \mathbf{y}$ *řešení* $\mathbf{x}$ *splňující*

$$\|\mathbf{x}\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{A})}\right), \tag{1.15}$$

pak $\mathbf{x}$ *je nutně nejřidší možné a je jediné takové. Navíc tohoto řešení lze dosáhnout* ℓ_1 *minimalizací.*

Některé další známé podmínky ekvivalence ℓ_0 a ℓ_1 jsou k nalezení v [28].

2 Metody separace pro jednokanálová data

V této kapitole bude rozebráno především teoretické odvození a zpracování jednotlivých separačních metod. Tato práce se zaměřuje především na dvě základní metody a jejich rozšíření. První metoda se nazývá *Principal Component Pursuit (PCP)* a vychází z robustní analýzy hlavních komponent. PCP je konvexní metoda. Existuje ale také její nekonvexní varianta, které bude věnována jedna podkapitola.

Dále se jedná o *metodu dynamických modů (Dynamic Mode Decomposition – DMD)* a o její rozšíření, které vychází ze základní metody DMD a používá základní myšlenku vlnkové transformace. Jedná se o tzv. *Multiresolution DMD – mrDMD.*

Protože je vhodné výsledky výše zmíněných metod porovnat již se zaběhlými metodami, bude krátce vysvětlena jedna z jednodušších již zaběhlých metod. Jedná se o tzv. *mediánový filtr*. Jelikož tato metoda funguje díky speciálnímu uspořádání vstupních dat, je potřeba nejprve ukázat, jak tato data správně předzpracovat.

2.1 Předzpracování dat

V této práci je předpokládáno, že vstupními daty je buď video nebo série $n \in \mathbb{N}$ po sobě jdoucích obrazů. Postup bude vysvětlen pouze pro sérii n obrazů, protože video je možné rozdělit na jednotlivé snímky a dále pracovat pouze s těmito snímky.

Dále je předpokládáno, že se jedná o klasické 8bitové obrazy [41]. Tedy vezmemeli obraz pouze v odstínech šedi, každý pixel může nabývat jedné z 256 možných hodnot, které se pohybují od 0 (černá) po 255 (bílá). Budou-li vstupní snímky barevné, předpokládáme, že náleží do RGB prostoru. RGB barevný model je aditivní barevný model, ve kterém se barvy vytváří přidáváním různých množství červeného, zeleného a modrého světla. Pro 8bitový barevný obraz v RGB prostoru to tedy znamená, že každý pixel jednoho barevného kanálu může nabývat 256 různých hodnot (od 0 do 255).

Nejprve se zaměříme na data ve stupních šedi. Obraz ve stupních šedi lze reprezentovat jako matici s rozměry odpovídajícími rozlišení daného obrazu (počtu pixelů). Tato matice má pouze celočíselné hodnoty, protože jednotlivé pixely nabývají pouze celočíselných hodnot z intervalu $\langle 0, 255 \rangle$. Nyní vezmeme matici odpovídající prvnímu snímku série a označíme ji $\mathbf{M}_1$. Poté provedeme její vektorizaci (poznámka 1.5), čímž vznikne dlouhý sloupcový vektor reprezentující první snímek. Vektorizací zbylých snímků $\mathbf{M}_2, \mathbf{M}_3, \ldots, \mathbf{M}_n$ získáme matici vstupních dat $\mathbf{M}$, jejíž sloupce tvoří vektory jednotlivých snímků: $\mathbf{M} = [\operatorname{vec} \mathbf{M}_1, \operatorname{vec} \mathbf{M}_2, \ldots, \operatorname{vec} \mathbf{M}_n]$.

Pokud budou vstupní data barevná (např. v RGB), použijeme výše zmíněný postup pro každou barevnou složku zvlášť. Výsledkem budou 3 matice $\mathbf{M}^{\mathrm{R}}$, $\mathbf{M}^{\mathrm{G}}$, $\mathbf{M}^{\mathrm{B}}$, kde jednotlivé sloupce $\mathbf{M}^{\mathrm{R}}$ odpovídají vektorům červené složky jednotlivých

snímků. Stejný princip platí pro zelenou a modrou složku. Metodami separace pro barevná vstupní data se bude zabývat kapitola 3.

Budou-li všechny snímky stejné, vzniklá matice **M** bude mít všechny sloupce stejné – bude mít hodnost jedna. Pokud budou snímky odlišné, ale budou se lišit pouze v malém množství pixelů, které budou reprezentovat pohybující se objekt (např. auto jedoucí po silnici), budou i jednotlivé sloupce velmi podobné a lišit se budou pouze v pixelech odpovídajících danému pohybujícímu se objektu. Díky tomu je možné odvodit formulaci separace statické složky (pozadí) od dynamické (pohybující se) složky.

Matici **M** chceme vyjádřit jako součet dvou matic, které označíme **L** a **S**. Matice **L** by měla reprezentovat statické pozadí, tzn. měla by mít všechny sloupce identické. Matice **S** by měla mít nenulové hodnoty pouze na místech, kde se pohyboval objekt, tzn. na místech, kde se hodnota v daném sloupci lišila od hodnot většiny ostatních sloupců. Na zbylých místech by měla obsahovat pouze nuly.

V této kapitole budou metody separace nejprve vysvětleny na snímcích ve stupních šedi. V další kapitole bude provedeno rozšíření těchto metod pro barevné snímky v RGB prostoru.

2.2 Mediánový filtr

Mediánový filtr je společně s metodami jako průměrový filtr, min-max filtr, atd. jednou z nejstarších a nejčastěji používaných metod pro separaci videa. Přehled těchto základních metod je k nalezení např. v článcích [26], [8].

Medián je hodnota, která dělí řadu vzestupně seřazených dat na dvě stejně početné poloviny. Toho může být využito, jelikož předpokládáme, že sloupce matice **M** jsou podobné a liší se pouze v několika místech odpovídajících složce **S**. Předpokládejme, že všechny pixely, které se v čase nemění, patří do pozadí. Dále předpokládejme, že jednotlivé pixely mají v nadpoloviční většině snímků stejnou hodnotu. Pokud vezmeme první řádek matice **M**, získáme hodnoty prvního pixelu jednotlivých snímků. Tedy spočítáme-li medián prvního řádku matice **M**, získáme aproximaci hodnoty prvního pixelu pozadí. Obraz, který získáme výpočtem mediánů hodnot jednotlivých pixelů všech obrazů, považujeme za dostatečně dobrý odhad pozadí **L**. Dynamickou složku **S** spočítáme jako rozdíl jednotlivých obrazů a pozadí $\mathbf{S} = \mathbf{M} - \mathbf{L}$.

Hlavní výhodou mediánového filtru je odolnost vůči extrémním hodnotám, která vychází přímo z vlastností mediánu. Navíc se jedná o nelineární metodu, je tedy na rozdíl od lineárních metod jako je průměrový filtr schopna lépe zpracovat data obsahující odlehlé hodnoty. Další výhodou je velmi jednoduchá implementace, která téměř ve všech programovacích jazycích zabere jen pár řádků.

Bohužel má i velké omezení: Obsahují-li vstupní data „příliš mnoho“ pohybujících se objektů, může nastat, že počet obrázků obsahujících dynamickou složku na určitém místě výrazně přeroste počet obrázků obsahujících pozadí na tomto místě. V takové případě s velkou pravděpodobností medián získaný pro jednotlivé řádky již nebude odpovídat hodnotám pozadí. Tedy získaný odhad pozadí bude silně zkreslený. Z tohoto důvodu se mediánový filtr v praxi používá většinou dohromady s dalšími algoritmy počítačového vidění (např. pro rozpoznávání, detekci a sledování objektů) pro zlepšení jejich robustnosti.

2.3 Metoda Principal Component Pursuit (PCP)

Metoda *Principal Component Pursuit (PCP)* vychází z tzv. *robustní analýzy hlavních komponent (Robust Pricnipal Component Analysis – RPCA)*. RPCA je robustní verze klasické analýzy hlavních komponent (PCA) [3], což je statistická metoda užívaná pro identifikaci vzorců chování dat. PCA identifikuje v datech vzorce chování tak, že převede původní soubor proměnných na nový soubor již nekorelovaných proměnných, které se nazývají tzv. *hlavní komponenty*. Soubor hlavních komponent je vybrán tak, aby měl co nejmenší dimenzi a zároveň aby maximálně vystihoval chování původních dat.

Nejprve bude ukázáno, jak je možné klasické PCA aplikovat na problém separace pozadí ve videu.

2.3.1 PCA pro separaci videa

Jak aplikovat PCA na problém separace videa, je ukázáno v článcích [27], [2]. Metoda funguje na principu nalezení tzv. *eigenbackground* – jedná se podprostor sloužící k odhadu statického pozadí jednotlivých obrazů. Pro vygenerování tohoto podprostoru se nejprve z n vstupních snímků vypočítá průměrný snímek a kovarianční matice $\mathbf{C}$. Použijeme základní princip PCA [3] a danou kovarianční matici diagonalizujeme pomocí rozkladu na vlastní vektory a vlastní čísla:

$$\mathbf{L} = \Phi \mathbf{C} \Phi^T, \tag{2.1}$$

kde Φ je matice vlastních vektorů kovarianční matice $\mathbf{C}$ a $\mathbf{L}$ je diagonální matice příslušných vlastních čísel.

Dále je kvůli zmenšení dimenze dat ponecháno pouze N největších vlastních čísel $\mathbf{L}_N$ a jim odpovídající vlastní vektory Φ_N – ty tvoří N-dimenzionální podprostor, který považujeme za hledané eigenbackground.

Každý vstupní snímek $\mathbf{M}_i$ může být poté projektován do podprostoru Φ_N, čímž jsou modelovány statické složky scény daného obrazu – označíme je $\mathbf{B}_i$. Spočítáme-li

eukleidovskou vzdálenost rozdílu vstupního snímku $\mathbf{M}_i$ a jeho modelu pozadí $\mathbf{B}_i$, můžeme pomocí prahu T detekovat, kde se ve snímku nacházejí pohybující se objekty

$$\|\mathbf{M}_i - \mathbf{B}_i\|_2 > T. \tag{2.2}$$

Použitím dané PCA metody získáme odhad pozadí, ale nezjistíme žádné informace o pohybujících se objektech. Z toho vyplývá první omezení metody – dynamická složka musí být malá a nesmí se objevovat na stejných místech. Jinak může nastat, že dané pohybující se objekty budou přidány do pozadí. Další nevýhodou je velká výpočetní náročnost potřebná pro přepočítávání modelu pozadí vzhledem k novým vstupním datům.

Z těchto důvodů byla snaha PCA metodu zrobustnit, aby si byla schopna poradit i s extrémními případy dynamické složky, čímž vznikla robustní analýza hlavních komponent (RPCA).

2.3.2 RPCA pro separaci videa

Tato a následující podkapitoly zabývající se konvexní PCP metodou jsou částečně převzaty a dále rozšiřují bakalářskou práci [10]. Následující teorie vychází z článků [4], [12].

RPCA na rozdíl od PCA neklade kromě řídkosti žádné další podmínky, jak by měla vypadat dynamická složka. Zatímco pro PCA musí být dynamická složka relativně malá, při RPCA je předpokládáno, že je dynamická složka pouze řídká. Nenulové hodnoty této složky mohou být ovšem jakkoliv velké.

Předpoklad pouhé řídkosti dynamické složky je užitečný, protože jak bylo ukázáno v podkapitole 2.1, matice vstupních dat $\mathbf{M}$ lze vyjádřit jako součet řídké matice $\mathbf{S}$ s maticí $\mathbf{L}$, která má v ideálním případě všechny sloupce identické – tzn. rank $\mathbf{L} = 1$. V reálném případě bude hodnost vyšší, ale stále poměrně malá, proto budeme matici $\mathbf{L}$ označovat jako *nízkohodnostní*.

Cílem metody je tyto dvě matice odseparovat. Zapíšeme-li daný problém formálně, získáme optimalizační úlohu

$$\min \ \mathrm{rank}(\mathbf{L}) + \|\mathbf{S}\|_0 \ \text{ za podmínky } \ \mathbf{L} + \mathbf{S} = \mathbf{M}. \tag{2.3}$$

Počet nenulových singulárních čísel odpovídá hodnosti matice. Tudíž chceme-li, aby matice $\mathbf{L}$ měla co nejmenší hodnost, musíme minimalizovat počet nenulových singulárních čísel dané matice. Toho lze ve většině případů dosáhnout pomocí nukleární normy, protože nukleární norma určuje hodnotu součtu singulárních čísel dané matice.

Norma ℓ_0 ukazuje, kolik je v matici nenulových prvků. Tedy, aby byla matice co nejřidší, musíme minimalizovat její ℓ_0 normu. Jak už bylo řečeno v podkapitole

1.6, norma ℓ_0 není konvexní problém. Z tohoto důvodu se pokusíme úlohu (2.3) přeformulovat pomocí ℓ_1 normy a nukleární normy [4]:

$$\min \ \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 \ \text{ za podmínky } \ \mathbf{L} + \mathbf{S} = \mathbf{M}. \tag{2.4}$$

Úloha (2.4) je již konvexní, tudíž jsme ji schopni řešit pomocí metod konvexní optimalizace. Nyní je potřeba zmínit podmínky, které musí vstupní data splňovat, aby bylo možné přesně odseparovat složky $\mathbf{L}$ a $\mathbf{S}$. Pokud budou dané podmínky splněny, měli bychom být schopni (2.4) vyřešit, i když hodnost matice $\mathbf{L}$ poroste téměř lineárně s dimenzí $\mathbf{M}$ a počet nenulových čísel matice $\mathbf{S}$ bude odpovídat zlomku všech prvků $\mathbf{M}$.

Předpoklady

Na první pohled se zdá, že pro vyřešení úlohy (2.4) musí matice $\mathbf{M}$ splňovat mnoho předpokladů. Podle [4] tomu tak vůbec není. I když nebudeme znát počet pohybujících se objektů nebo jejich pravděpodobný výskyt, stále můžeme provést přesnou dekompozici. Stačí, když zajistíme, aby $\mathbf{M}$ nebyla nízkohodnostní a řídká zároveň.

Tedy prvně je potřeba zajistit, aby nízkohodnostní prvek $\mathbf{L}$ nebyl zároveň řídký. K tomu budou použity tzv. *podmínky inkoherence*. Provedeme-li singulární rozklad $\mathbf{L} \in \mathbb{R}^{m\times n}$.

$$\mathbf{L} = \mathbf{U}\Sigma\mathbf{V}^*$$

Potom podmínky inkoherence určené parametrem μ zní

$$\max_i \|\mathbf{U}^*\mathbf{e}_i\|^2 \le \frac{\mu r}{m}, \quad \max_i \|\mathbf{V}^*\mathbf{e}_i\|^2 \le \frac{\mu r}{n}, \tag{2.5}$$

$$\|\mathbf{U}\mathbf{V}^*\|_\infty \le \sqrt{\frac{\mu r}{mn}}, \tag{2.6}$$

kde r je hodnost matice $\mathbf{L}$ a $\mathbf{e}_i$ je jednotkový vektor.

Podmínky (2.5) a (2.6) zaručují, že pro malé hodnoty μ budou singulární vektory $\mathbf{L}$ vhodně rozmístěné – jinými slovy matice $\mathbf{L}$ není řídká.

Další problém může nastat, pokud řídká matice $\mathbf{S}$ bude zároveň i nízkohodnostní. Proto budeme po řídké matici požadovat, aby její nenulové prvky byly rovnoměrně rozmístěné, jak bylo zmíněno v podkapitole 1.2.

Principal Component Pursuit – PCP

Pokud jsou splněny výše zmíněné základní předpoklady (hodnost $\mathbf{L}$ není příliš velká a prvek $\mathbf{S}$ je rozumně řídký), platí:

Tvrzení 2.1. [4] *Nechť $\mathbf{L}_{n,n}$ splňuje podmínky* (2.5),(2.6). *Pevně zvolme znaménkovou matici $\Sigma_{n,n}$ a předpokládejme, že nosič Ω matice $\mathbf{S}$ je rovnoměrně rozdělen*

mezi všemi množinami o mohutnosti k a že $\operatorname{sgn}([\mathbf{S}]_{ij}) = \Sigma_{ij}$ *pro všechny* $(i,j) \in \Omega$. *Pak existuje konstanta* $c \in \mathbb{R}$ *taková, že s pravděpodobností alespoň* $1 - cn^{-10}$ *je řešení úlohy* (2.4)*, které nazveme Principal Component Pursuit (PCP), s* $\lambda = 1/\sqrt{n}$ *přesné. Tzn.* $\hat{\mathbf{L}} = \mathbf{L}$ *a* $\hat{\mathbf{S}} = \mathbf{S}$ *za předpokladu, že*

$$\operatorname{rank}(\mathbf{L}) \leq \rho_r n \mu^{-1} (\log\, n)^{-2} \quad \text{a} \quad k \leq \rho_s n^2,$$

kde $\hat{\mathbf{L}}$, $\hat{\mathbf{S}}$ *jsou matice získané separací a* ρ_r, ρ_s *jsou kladné konstanty. V obecném případě, kdy* $\mathbf{L}_{m,n}$ *je obdélníková, PCP s* $\lambda = 1/\sqrt{n_{\max}}$ *uspěje s pravděpodobností alespoň* $1 - cn_{\max}^{-10}$*, pokud*

$$\operatorname{rank}(\mathbf{L}) \leq \rho_r n_{\min} \mu^{-1} (\log\, n_{\max})^{-2} \quad \text{a} \quad k \leq \rho_s mn,$$

kde $n_{\max} = \max(m,n)$ *a* $n_{\min} = \min(m,n)$.

Tedy matice $\mathbf{L}$, jejíž singulární vektory (hlavní komponenty) jsou rozumně rozložené, může být z dat se zcela neznámým výskytem pohybujících se objektů (pokud jsou náhodně rozložené) získána přesně s pravděpodobností téměř jedna. Toto platí dokonce i pro $\mathbf{L}$ s velkými hodnostmi (řádu $n/(\log\, n)^2$), pokud μ není příliš velké.

Výhodou PCP algoritmu je přítomnost univerzálního ladícího parametru λ. Dosadíme-li při splnění předpokladů tvrzení 2.1 $\lambda = 1/\sqrt{n_{\max}}$ do úlohy (2.4), úloha zkonverguje ke správnému výsledku separace. Máme tedy univerzální konstantu, která funguje nezávisle na tom, jaké jsou matice $\mathbf{L}$ a $\mathbf{S}$. Správnost volby λ je možné ověřit v důkazu tvrzení 2.1, který je k dispozici v článku [4]. Z tohoto důkazu navíc vyplývá, že $\lambda = 1/\sqrt{n_{\max}}$ není jediná vhodná volba λ. Experimentálním nalezením ideálního parametru λ pro konkrétní úlohy se budou zabývat další kapitoly.

2.3.3 Algoritmus PCP

Existuje více možných algoritmů pro řešení separace nízkohodnostní a řídké matice. Tato práce se zaměří na metodu využívající tzv. *rozšířený Lagrangeův multiplikátor (Augmented Lagrange Multiplier – ALM)*. ALM je využit, protože na rozdíl od starších metod, jako je iterativní prahování nebo metoda akcelerovaného proximálního gradientu, dosahuje rychlejších a přesnějších výsledků. Detailnější informace ohledně daných metod jsou dostupné v článku [24].

Stále existuje značné množství algoritmů využívajících ALM. Příkladem jsou *exact ALM* a rychlejší *inexact ALM* (dostupné v [24]). V článku [37] je ukázáno, že metoda *Alternating Direction Method of Multipliers – ADMM* konverguje rychleji než inexact ALM. Navíc by metoda měla být odolnější vůči šumu a odlehlým hodnotám. Proto i v této práci bude jako algoritmus pro vyřešení úlohy (2.4) použita právě metoda ADMM.

Alternating Direction Method of Multipliers (ADMM)

Tato podkapitola vychází z článků [38] a [37].

ADMM je metoda pro řešení problémů typu

$$\min_{\mathbf{x},\mathbf{z}} \ f(\mathbf{x}) + g(\mathbf{z}) \ \text{ za podmínky } \quad A\mathbf{x} - \mathbf{z} = \mathbf{0}, \tag{2.7}$$

kde $\mathbf{x} \in \mathbb{C}^n$, funkce f, g jsou reálné a konvexní a $A : \mathbb{C}^n \to \mathbb{C}^p$ je lineární operátor. Pro vyřešení použijeme rozšířený Lagrangian příslušný výše zmíněné úloze (2.7):

$$L_\rho(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^\top(A\mathbf{x} - \mathbf{z}) + \frac{\rho}{2}\|A\mathbf{x} - \mathbf{z}\|_2^2, \tag{2.8}$$

přičemž $\rho > 0$ je penalizační parametr.

Upravíme-li poslední dva sčítance ve vztahu (2.8) dosazením rezidua $\mathbf{r} = A\mathbf{x} - \mathbf{z}$, dostaneme tzv. *škálovaný tvar rozšířeného Lagrangianu* (celé odvození je k nalezení v [38])

$$L_\rho(\mathbf{x}, \mathbf{u}, \mathbf{z}) = f(\mathbf{x}) + g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{r} + \mathbf{u}\|_2^2 - \frac{\rho}{2}\|\mathbf{u}\|_2^2, \tag{2.9}$$

kde $\mathbf{u} = \mathbf{y}/\rho$.

Potom škálovaná verze ADMM pro problém (2.7) vypadá jako následující iterační úloha [38].

$$\mathbf{x}^{(i+1)} = \arg\min_{\mathbf{x}} \ \left(f(\mathbf{x}) + \frac{\rho}{2}\|A\mathbf{x} - \mathbf{z}^{(i)} + \mathbf{u}^{(i)}\|_2^2 \right) \tag{2.10}$$

$$\mathbf{z}^{(i+1)} = \arg\min_{\mathbf{z}} \ \left(g(\mathbf{z}) + \frac{\rho}{2}\|A\mathbf{x}^{(i+1)} - \mathbf{z} + \mathbf{u}^{(i)}\|_2^2 \right) \tag{2.11}$$

$$\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + A\mathbf{x}^{(i+1)} - \mathbf{z}^{(i+1)} \tag{2.12}$$

V (2.10) bylo možno vynechat člen $g(\mathbf{z})$, protože nezávisí na $\mathbf{x}$. Ze stejného důvodu můžeme vynechat člen $\frac{\rho}{2}\|\mathbf{u}\|_2^2$. Pro výraz (2.11) platí podobná argumentace.

2.3.4 Proximální operátory

Podkapitola vychází z [28], [9]. Výrazy (2.10) a (2.11) nelze řešit přímočaře. Proto se musíme, dříve než přejdeme k samotnému algoritmu pro PCP problém (2.4), seznámit s proximálními operátory. Tyto operátory se někdy používají k řešení problémů konvexní optimalizace.

Definice 2.2. *Jestliže úloha*

$$\arg\min_{\mathbf{y}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 + f(\mathbf{y}) \tag{2.13}$$

má pro danou funkci f jednoznačné řešení, pak toto řešení značíme $\text{prox}_f\mathbf{x}$ *a operátor* $\text{prox}_f : \mathbb{R}^n \to \mathbb{R}^n$ *nazýváme proximálním operátorem funkce f.*

Poznámka 2.3. *Řešení úlohy* (2.13) *je jednoznačné, je-li $f : \mathbb{R}^n \to (-\infty, \infty)$ zdola polospojitá konvexní funkce s neprázdným definičním oborem* [28].

Proximální operátor ℓ_1 normy

Jak už bylo výše uvedeno, k nalezení řídké matice $\mathbf{S}$ bude využita ℓ_1 norma. Jelikož v úloze (2.4) vystupuje ℓ_1 norma společně s regularizačním parametrem λ, uvedeme proximální operátor pro případ $f = \lambda\|\mathbf{y}\|_1$ [9]. Dosazením do definice (2.13) dostáváme

$$\mathrm{prox}_{\lambda\|\cdot\|_1}(\mathbf{x}) = \arg\min_{\mathbf{y}} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + \lambda\|\mathbf{y}\|_1$$

a rozepíšeme-li normy po složkách, získáme

$$\mathrm{prox}_{\lambda\|\cdot\|_1}(\mathbf{x}) = \arg\min_{\mathbf{y}} g(y), \quad g(y) = \frac{1}{2}\sum_{i=1}^{n}(x_i - y_i)^2 + \lambda\sum_{i=1}^{n}|y_i|.$$

Funkce $g(y)$ je konvexní a parciálně diferencovatelná podle y_i všude kromě bodu $y_i = 0$. Její minimum nalezneme tak, že spočítáme její derivaci a položíme ji rovnu nule. Postupujeme-li takto na jednotlivých intervalech [9], dojdeme k výsledku

$$y_i = \frac{x_i}{|x_i|}\max(|x_i| - \lambda, 0) \tag{2.14}$$

platícímu pro všechna nenulová x_i. Funkci (2.14) nazveme *měkké prahování* nebo také *soft thresholding* a označíme ji

$$\mathrm{soft} = y_i = \frac{x_i}{|x_i|}\max(|x_i| - \lambda, 0). \tag{2.15}$$

Proximální operátor nukleární normy

Druhou normou, která se v úloze (2.4) vyskytuje, je nukleární norma. Jak bylo dříve řečeno, nukleární normu je možné nahradit ℓ_1 normou singulárních čísel. Využijeme již získané funkce (2.14) pro ℓ_1 normu, čímž získáme operátor nukleární normy

$$\mathrm{svt}_\lambda(\mathbf{X}) = \mathrm{prox}_{\lambda\|\cdot\|_*}(\mathbf{X}) = \sum_{i=1}^{n}\mathrm{soft}_\lambda(\sigma_i)\mathbf{u}_i\mathbf{v}^*{}_i, \tag{2.16}$$

který nazveme *singular value thresholding – SVT.*

Odvození algoritmu PCP

Nyní stačí odvodit algoritmus přímo pro úlohu PCP. K tomu využijeme již dříve zmíněný rozšířený Lagrangian. Ten pro problém (2.4) vypadá následovně [4]

$$L_\mu(\mathbf{L},\mathbf{S},\mathbf{Y}) = \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 + \langle\mathbf{Y}, \mathbf{M}-\mathbf{L}-\mathbf{S}\rangle + \frac{\mu}{2}\|\mathbf{M}-\mathbf{L}-\mathbf{S}\|_F^2, \tag{2.17}$$

kde $\mathbf{Y}$ je tzv. *Lagrangeův multiplikátor.*
Nyní úpravou posledních dvou členů vztahu (2.17) získáme jejich škálovaný tvar.

$$\langle\mathbf{Y}, \mathbf{M}-\mathbf{L}-\mathbf{S}\rangle + \frac{\mu}{2}\|\mathbf{M}-\mathbf{L}-\mathbf{S}\|_F^2 = \frac{\mu}{2}\|\mathbf{M}-\mathbf{L}-\mathbf{S}+\mathbf{Y}/\mu\|_2 - \frac{\mu}{2}\|\mathbf{Y}/\mu\|_2. \tag{2.18}$$

Bez újmy na obecnosti předpokládejme, že $\mathbf{M},\ \mathbf{L},\ \mathbf{S}$ a $\mathbf{Y}$ jsou vektory (matice lze vektorizovat). Označíme-li $\mathbf{R} = \mathbf{M} - \mathbf{L} - \mathbf{S}$, tak platí

$$\begin{aligned}
&\frac{\mu}{2}\|\mathbf{R} + \mathbf{Y}/\mu\|_2 - \frac{\mu}{2}\|\mathbf{Y}/\mu\|_2 = \frac{\mu}{2}\langle \mathbf{R} + \mathbf{Y}/\mu, \mathbf{R} + \mathbf{Y}/\mu\rangle - \frac{\mu}{2}\langle \mathbf{Y}/\mu, \mathbf{Y}/\mu\rangle \\
&= \frac{\mu}{2}\left(\langle \mathbf{R}, \mathbf{R}\rangle + \langle \mathbf{R}, \mathbf{Y}/\mu\rangle + \langle \mathbf{Y}/\mu, \mathbf{R}\rangle + \langle \mathbf{Y}/\mu, \mathbf{Y}/\mu\rangle\right) - \frac{\mu}{2}\langle \mathbf{Y}/\mu, \mathbf{Y}/\mu\rangle \\
&= \frac{\mu}{2}\left(\|\mathbf{R}\|_2^2 + \frac{2}{\mu}\langle \mathbf{R}, \mathbf{Y}\rangle\right) = \frac{\mu}{2}\|\mathbf{R}\|_2^2 + \langle \mathbf{R}, \mathbf{Y}\rangle.
\end{aligned}$$

Předposlední úprava je korektní, protože pro skalární součin v reálném oboru přímo z definice platí $\langle u, v\rangle = \langle v, u\rangle$ a $\langle \lambda u, v\rangle = \lambda\langle u, v\rangle$. Jelikož Frobeniova norma odpovídá ℓ_2 normě, vztah (2.18) je korektní.

Škálovaný tvar daného rozšířeného Lagrangianu je následovný:

$$L_\mu(\mathbf{L}, \mathbf{S}, \mathbf{Y}) = \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 + \frac{\mu}{2}\|\mathbf{M} - \mathbf{L} - \mathbf{S} + \mathbf{Y}/\mu\|_2 - \frac{\mu}{2}\|\mathbf{Y}/\mu\|_2. \tag{2.19}$$

Odtud získáme škálovaný tvar ADMM pro PCP problém (2.4)

$$\mathbf{L}^{(i+1)} = \arg\min_{\mathbf{L}} \left(\frac{\mu}{2}\|\mathbf{M} - \mathbf{L} - \mathbf{S}^{(i)} + \mathbf{Y}^{(i)}/\mu\|_2^2 + \|\mathbf{L}\|_*\right) \tag{2.20}$$

$$\mathbf{S}^{(i+1)} = \arg\min_{\mathbf{S}} \left(\frac{\mu}{2}\|\mathbf{M} - \mathbf{L}^{(i+1)} - \mathbf{S} + \mathbf{Y}^{(i)}/\mu\|_2^2 + \lambda\|\mathbf{S}\|_1\right) \tag{2.21}$$

$$\mathbf{Y}^{(i+1)} = \mathbf{Y}^{(i)} + \mu\left(\mathbf{M} - \mathbf{L}^{(i+1)} - \mathbf{S}^{(i+1)}\right). \tag{2.22}$$

Všimněme si, že výraz (2.20) připomíná proximální operátor nukleární normy

$$\mathrm{prox}_{\|\cdot\|_*}(\mathbf{x}) = \arg\min_{\mathbf{y}} \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 + \|\mathbf{y}\|_*. \tag{2.23}$$

Dosazením $\mathbf{y} = \mathbf{L}$ a $\mathbf{x} = \mathbf{M} - \mathbf{S}^{(i)} + \mathbf{Y}^{(i)}/\mu$ do vztahu (2.23) získáme tvar

$$\mathrm{prox}_{\|\cdot\|_*}(\mathbf{M} - \mathbf{S}^{(i)} + \mathbf{Y}^{(i)}/\mu) = \arg\min_{\mathbf{L}} \frac{1}{2}\|\mathbf{M} - \mathbf{L} - \mathbf{S}^{(i)} + \mathbf{Y}^{(i)}/\mu\|_2^2 + \|\mathbf{L}\|_*. \tag{2.24}$$

Vydělíme-li výraz (2.20) konstantou μ, dostaneme stejný tvar jako v rovnici (2.24). Toto dělení můžeme provést, protože se změní pouze hodnota minima, ale výsledek zůstane nezměněn. Použitím rovnice (2.16) platí

$$\mathbf{L}^{(i+1)} = \mathrm{svt}_{\frac{1}{\mu}}\left(\mathbf{M} - \mathbf{S}^{(i)} + \mathbf{Y}^{(i)}/\mu\right). \tag{2.25}$$

Výraz (2.21) tentokrát připomíná proximální operátor ℓ_1 normy

$$\mathrm{prox}_{\lambda\|\cdot\|_1}(\mathbf{x}) = \arg\min_{\mathbf{y}} \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{y}\|_1. \tag{2.26}$$

Obdobně dosazením $\mathbf{y} = \mathbf{S}$ a $\mathbf{x} = \mathbf{M} - \mathbf{L}^{(i+1)} + \mathbf{Y}^{(i)}/\mu$ do vztahu (2.26) a vydělením výrazu (2.21) konstantou μ, dostáváme stejné tvary. Použijeme-li rovnici (2.15), tak platí

$$\mathbf{S}^{(i+1)} = \mathrm{soft}_{\frac{\lambda}{\mu}}\left(\mathbf{M} - \mathbf{L}^{(i+1)} + \mathbf{Y}^{(i)}/\mu\right). \tag{2.27}$$

Pomocí výše odvozených výrazů získáváme algoritmus PCP [4].

Algoritmus 1: Principal Component Pursuit (PCP)

inicializace: $\mathbf{S}_0 = \mathbf{Y}_0 = 0$, $\mu > 0$;
while $\|\mathbf{M} - \mathbf{L}_{k+1} - \mathbf{S}_{k+1}\|_F > \delta\|\mathbf{M}\|_F$ **do**
  $\mathbf{L}_{k+1} = \mathrm{svt}_{\frac{1}{\mu}}(\mathbf{M} - \mathbf{S}_k + \mathbf{Y}_k/\mu)$;
  $\mathbf{S}_{k+1} = \mathrm{soft}_{\frac{\lambda}{\mu}}(\mathbf{M} - \mathbf{L}_{k+1} + \mathbf{Y}_k/\mu)$;
  $\mathbf{Y}_{k+1} = \mathbf{Y}_k + \mu(\mathbf{M} - \mathbf{L}_{k+1} - \mathbf{S}_{k+1})$;
end
Výsledek: $\mathbf{L}, \mathbf{S}$

Parametr δ v algoritmu 1 udává, při jaké dosažené přesnosti se algoritmus zastaví. Jak tento parametr zvolit bude rozebráno v dalších kapitolách.

2.4 Nekonvexní RPCA

Jak bylo ukázáno v minulé kapitole, RPCA formuluje problém separace jako

$$\min\ \mathrm{rank}(\mathbf{L}) + \|\mathbf{S}\|_0 \ \text{ za podmínky } \ \mathbf{L} + \mathbf{S} = \mathbf{M}. \tag{2.28}$$

Minimalizaci hodnosti matice $\mathbf{L}$ je možné z důvodu zjednodušení výpočtu převést na minimalizaci nukleární normy $\mathbf{L}$. Protože ℓ_0 norma je nekonvexní problém, byla v PCP metodě tato norma převedena na již konvexní ℓ_1 normu. Takto vzniklou úlohu už jsme schopni rozumně vyřešit, ale za cenu méně přesné separace.

V ideálním případě bychom tedy chtěli být schopni vyřešit přímo úlohu

$$\min\ \|\mathbf{L}\|_* + \|\mathbf{S}\|_0 \ \text{ za podmínky } \ \mathbf{L} + \mathbf{S} = \mathbf{M}. \tag{2.29}$$

Pokud není možné spočítat přímo ℓ_0 normu, bylo by vhodné vyzkoušet jiné nekonvexní normy ℓ_p, kde $p \in (0, 1)$. S řešením tohoto problému pro ℓ_p, kde $p \in \langle 0, 1)$, přišel R. Chartrand v článku [6]. V tomto článku navíc představil možnost regularizace řídké složky, která je užitečná, jsou-li vstupní data silně zašuměná.

Nejdříve je potřeba objasnit základní matematické pojmy, které jsou použity k formulaci nekonvexního algoritmu RPCA úlohy.

2.4.1 Základní pojmy

Huberova funkce

Tuto funkci poprvé představil P. J. Huber v roce 1964 v článku [15]. Jedná se o účelovou funkci používanou k regresní analýze převážně pro svoji odolnost vůči zašuměným datům a odlehlým hodnotám.

Huberovu funkci značíme $h_\mu(x)$ a definujeme následovně

$$h_\mu(x) = \begin{cases} \frac{|x|^2}{2\mu} & |x| \leq \mu \\ |x| - \frac{\mu}{2} & |x| \geq \mu. \end{cases} \tag{2.30}$$

Funkce je kvadratická pro malé hodnoty x, čímž je více robustní vůči šumu, a lineární pro větší hodnoty x, čímž je více robustní vůči odlehlým pozorováním. Parametr μ potom určuje práh, kdy je funkce lineární a kdy kvadratická.

Moreauova obálka

Moreauova obálka ([36], [1]) je nástroj často používaný v optimalizaci, protože pomocí Moreauovy obálky lze získat k nehladké funkci její hladkou aproximaci.

Definice 2.4. [36] *Moreauva obálka funkce $f : \mathscr{H} \to \mathbb{R}$ s parametrem β je funkce f^β definovaná jako*

$$f^\beta(x) = \inf_{y \in \mathscr{H}} \{\frac{1}{2\beta}\|y - x\|^2 + f(y)\}, \tag{2.31}$$

kde $\mathscr{H}$ značí Hilbertův prostor.

Moreauova obálka má množství užitečných vlastností – je reálná, konvexní a spojitá. Jak již bylo řečeno, f^β je hladkou aproximací f. Navíc čím je parametr β menší, tím je Moreauova obálka f^β blíže k původní funkci f. Tedy f^β konverguje k f pro $\beta \to 0^+$. Důkaz, že zmíněné vlastnosti platí, je k nalezení v knize [1].

Legendre–Fenchelova transformace

Legendre–Fenchelova transformace ([1], [6]) je matematická operace, která zobrazuje funkci na její konvexní sdruženou funkci. Tato transformace zobrazuje funkci z jejího původního definičního oboru do duálního prostoru, kde jsou role proměnných obráceny.

Používá se zejména v konvexní analýze a optimalizaci, protože poskytuje způsob, jak transformovat nekonvexní funkce do jejich konvexních konjugací, se kterými je často snazší pracovat.

Definice 2.5. [6] *Je dána funkce $f : \mathbb{R}^n \to \mathbb{R}$, její Legendre–Fenchelova transformace $f^* : \mathbb{R}^n \to \bar{\mathbb{R}}$ je reálná funkce rozšířená o $\{-\infty, \infty\}$ definovaná jako*

$$f^* = \max_x x \cdot y - f(x) \tag{2.32}$$

a bikonjugace f je $f^{**} = (f^*)^*$.

Zajímavostí této transformace je, že pro spojitou f platí, že je rovna svojí bikonjugaci $f = f^{**}$ ([6]).

2.4.2 Odvození nekonvexní metody

Následující podkapitoly zabývající se nekonvexní metodou vychází převážně ze článku [6].

Konvexní případ ℓ_1

V této podkapitole se budeme blíže zabývat použitím konvexní ℓ_1 normy. Ukázané vztahy budou potřeba v další podkapitole 2.4.2, ve které bude ukázáno, že je možné tyto vztahy zobecnit tak, aby se daly použít i pro nekonvexní l_p normy.

Zjednodušíme-li PCP (konvexní) problém (2.4) tak, že zvolíme pevné $\mathbf{L}$ a podmínku $\mathbf{M} = \mathbf{L} + \mathbf{S}$ nahradíme výrazem zaručujícím věrnost dat, získáme jednodušší problém:

$$\mathbf{H}_{\mu,1}(\mathbf{T}) := \min_{\mathbf{S}} \|\mathbf{S}\|_1 + \frac{1}{2\mu}\|\mathbf{S} - \mathbf{T}\|_F^2, \tag{2.33}$$

kde $\mu > 0$. Jelikož je funkce $\mathbf{H}_{\mu,1}$ separabilní [6], můžeme úlohu (2.33) řešit zvlášť po složkách. Tedy $\mathbf{H}_{\mu,1}(\mathbf{T}) = \sum_{i,j} \mathbf{h}_{\mu,1}(t_{ij})$, přičemž

$$\mathbf{h}_{\mu,1}(\mathbf{t}) := \min_{\mathbf{s}} |\mathbf{s}| + \frac{1}{2\mu}|\mathbf{s} - \mathbf{t}|^2. \tag{2.34}$$

Funkce zadefinovaná pomocí (2.34) je Moreauovou obálkou funkce $|\cdot|$. Zároveň je $\mathbf{h}_{\mu,1}(\mathbf{t})$ proximálním operátorem $|\cdot|$, tudíž řešení úlohy (2.34) je již zmíněné měkké prahování

$$\mathbf{s}^* = \mathrm{soft}_\mu(\mathbf{t}) := \max\{0, |\mathbf{t}| - \mu\}\frac{\mathbf{t}}{|\mathbf{t}|}. \tag{2.35}$$

Dosadíme-li do (2.34) řešení (2.35) získané pro $\mathbf{s}$, dostaneme přímé řešení pro $\mathbf{t}$. Pro případ $|\mathbf{t}| \leq \mu$ je $\mathbf{s} = 0$, tedy $\mathbf{h}_{\mu,1}(\mathbf{t}) = \frac{1}{2\mu}|\mathbf{t}|^2$. Pokud je $|\mathbf{t}| \geq \mu > 0$, tak $\frac{\mathbf{t}}{|\mathbf{t}|} = 1$ a $\mathbf{h}_{\mu,1}(\mathbf{t}) = (|\mathbf{t}| - \mu)\frac{\mathbf{t}}{|\mathbf{t}|} + \frac{1}{2\mu}|(|\mathbf{t}| - \mu)\frac{\mathbf{t}}{|\mathbf{t}|} - \mathbf{t}|^2 = \mathbf{t} - \mu + \frac{1}{2\mu}|-\mu|^2 = \mathbf{t} - \frac{\mu}{2}$.

Tedy řešení funkce $\mathbf{t}$ definované pomocí (2.34) je

$$\mathbf{h}_{\mu,1}(\mathbf{t}) = \begin{cases} \frac{|\mathbf{t}|^2}{2\mu} & |\mathbf{t}| \leq \mu \\ |\mathbf{t}| - \frac{\mu}{2} & |\mathbf{t}| \geq \mu. \end{cases} \tag{2.36}$$

Ze vztahu (2.36) je rovněž viditelné, že toto řešení je Huberova funkce.

Zobecnění pro nekonvexní případy

Nyní bychom chtěli najít zobecnění (2.34), které by umožňovalo použití nekonvexní optimalizace a zároveň bylo stále efektivně řešitelné.

Způsob zobecnění problému pro nekonvexní případy, který by každého hned napadl, je použít namísto ℓ_1 normy nekonvexní ℓ_p, $0 < p < 1$ normu. Tento přístup byl zvolen v článku [18]. Bohužel řešení takového problému již nebude dáno pomocí měkkého prahování. Navíc by bylo možné toto řešení zapsat explicitně pouze pro vybrané hodnoty p. Z tohoto důvodu bude zvolen jiný přístup.

Zkonstruujme nekonvexní funkci $\mathbf{g}$ tak, že optimalizační problém

$$\min_{\mathbf{s}} \mathbf{g}(\mathbf{s}) + \frac{1}{2\mu}|\mathbf{s}-\mathbf{t}|^2 \tag{2.37}$$

lze vyřešit zobecněním měkkého prahování (2.35). Následně zobecníme Huberovu funkci:

$$\mathbf{h}_{\mu,p}(\mathbf{t}) = \begin{cases} \frac{|\mathbf{t}|^2}{2\mu} & |\mathbf{t}| \leq \mu^{\frac{1}{2-p}} \\ \frac{|\mathbf{t}|^p}{p} - \delta & |\mathbf{t}| \geq \mu^{\frac{1}{2-p}}, \end{cases} \tag{2.38}$$

kde $\delta = (\frac{1}{p} - \frac{1}{2})\mu^{\frac{p}{2-p}}$ je dopočítáno tak, aby $\mathbf{h}_{\mu,p}(\mathbf{t})$ byla spojitou funkcí se spojitými prvními derivacemi pro všechna $p \in \mathbb{R}$. Speciálně pro případ $p = 0$ dodefinujeme funkci tak, že $|\mathbf{t}|^p$ chápeme jako $\log(|\mathbf{t}|)$ a $\delta = \frac{\log\mu - 1}{2}$.

Nyní bychom potřebovali vyjádřit $\mathbf{h}_{\mu,p}$ jako Moreauovu obálku funkce $\mathbf{g}_{\mu,p}$, stejně jako je (2.34) Moreauovou obálkou (2.36) pro případ $p = 1$. Funkci $\mathbf{g}_{\mu,p}$ zadefinujeme jako

$$\frac{|\mathbf{s}|^2}{2} + \mu\mathbf{g}_{\mu,p}(\mathbf{s}) = \left(\frac{|\cdot|^2}{2} - \mu\mathbf{h}_{\mu,p}\right)^*(\mathbf{s}), \tag{2.39}$$

kde * značí Legendre–Fenchelovu transformaci.

Dále zkonstruujme funkci $\mathbf{f}_{\mu,p}(\mathbf{t}) := \frac{|\mathbf{t}|}{2} - \mu\mathbf{h}_{\mu,p}(\mathbf{t})$, která je pro $p \leq 1$ konvexní. Jelikož je $\mathbf{f}_{\mu,p}$ spojitá, je rovna svoji bikonjugaci

$$\mathbf{f}_{\mu,p}(\mathbf{t}) = \mathbf{f}^{**}_{\mu,p}(\mathbf{t}) = \left(\frac{|\cdot|^2}{2} + \mu\mathbf{g}_{\mu,p}\right)^*(\mathbf{t}) = \max_{\mathbf{s}} \mathbf{s}\cdot\mathbf{t} - \frac{|\mathbf{s}|^2}{2} - \mu\mathbf{g}_{\mu,p}(\mathbf{s}). \tag{2.40}$$

Z výše uvedeného vztahu $\mathbf{f}_{\mu,p} = \mathbf{f}^{**}_{\mu,p}$ získáváme rovnici

$$\max_{\mathbf{s}} \mathbf{s}\cdot\mathbf{t} - \frac{|\mathbf{s}|^2}{2} - \mu\mathbf{g}_{\mu,p}(\mathbf{s}) = \frac{|\mathbf{t}|}{2} - \mu\mathbf{h}_{\mu,p}(\mathbf{t}). \tag{2.41}$$

Vyjádřením $\mathbf{h}_{\mu,p}$ z této rovnice získáme požadovanou Moreauovu obálku funkce $\mathbf{g}_{\mu,p}$

$$\mathbf{h}_{\mu,p}(\mathbf{t}) := \min_{\mathbf{s}} \mathbf{g}_{\mu,p}(\mathbf{s}) + \frac{1}{2\mu}|\mathbf{s}-\mathbf{t}|^2. \tag{2.42}$$

Toto zobecnění bylo provedeno na vektorech. Nyní funkci $\mathbf{g}_{\mu,p}$ rozšíříme i pro matice.

Definice 2.6. [6] *Nechť je* $\mathbf{g}_{\mu,p}$ *dáno vztahem* (2.39)*, pak penalizační funkci*

$$\mathbf{G}_{\mu,p}(\mathbf{S}) := \sum_{prvky\ \mathbf{s}\ matice\ \mathbf{S}} \mathbf{g}_{\mu,p}(\mathbf{s}) \tag{2.43}$$

nazveme proximální p-norma matice $\mathbf{S}$*.*

$\mathbf{S}$ může být vektor, matice nebo vícerozměrné pole.

Poznámka 2.7. *I když jsme funkci* $\mathbf{G}_{\mu,p}$ *pojmenovali* proximální p-norma*, ve skutečnosti se jedná spíš o pseudonormu, protože nemusí splňovat předpoklad pozitivní homogenity. Ostatní předpoklady normy funkce splňuje, protože je* $\mathbf{G}_{\mu,p}$ *pro* $p \leq 1$ *radiální, rostoucí, nezáporná, nehladká, spojitá a splňuje trojúhelníkovou nerovnost. Důkaz daných vlastností je k dispozici v článku* [6]*.*

Nyní vyjádříme řešení proximálního operátoru získaného vztahem (2.42). Jedná se o *p-měkké prahování*

$$\mathbf{s}^* = \mathrm{soft}^p_\mu(\mathbf{t}) := \max\{0, |\mathbf{t}| - \mu|\mathbf{t}|^{p-1}\}\frac{\mathbf{t}}{|\mathbf{t}|}. \tag{2.44}$$

Z výše uvedených vztahů je zřejmé, že funkci $\mathbf{g}_{\mu,p}(\mathbf{t})$ nelze zapsat explicitně. Explicitní vyjádření je možné pouze pro několik vybraných hodnot p např. $p = \frac{1}{2}$. Tento fakt není omezující, protože stačí, že jsme schopni efektivně spočítat proximální operátor $\mathbf{g}_{\mu,p}$ pomocí p-měkkého prahování.

2.4.3 Nekonvexní ADMM algoritmus

Nyní je možné přeformulovat problém (2.4) na nekonvexní úlohu

$$\min_{\mathbf{L},\mathbf{S}} \ \mathbf{G}_{\mu,p}(\sigma(\mathbf{L})) + \lambda\mathbf{G}_{\mu,p}(\mathbf{S}) \ \text{ za podmínky } \ \mathbf{L} + \mathbf{S} = \mathbf{M}, \tag{2.45}$$

kde $\sigma(\mathbf{L})$ je vektor singulárních čísel matice $\mathbf{L}$.

Pro použití ADMM metody je potřeba získat rozšířený Lagrangian úlohy (2.45). Toho dosáhneme uvolněním podmínky $\mathbf{L} + \mathbf{S} = \mathbf{M}$ a přidáním Lagrangeova multiplikátoru $\mathbf{Y}$

$$\min_{\mathbf{L},\mathbf{S}} \ \mathbf{G}_{\mu,p}(\sigma(\mathbf{L})) + \lambda\mathbf{G}_{\mu\lambda,p}(\mathbf{S}) + \frac{1}{2\mu}\|\mathbf{M} - \mathbf{L} - \mathbf{S} - \mathbf{Y}\|^2_F. \tag{2.46}$$

Abychom získali jednotlivé kroky ADMM algoritmu pro daný Langrangian, postupně zvolíme jednotlivé proměnné pevně a vyřešíme úlohu pro zbylé.

Zvolme $\mathbf{L}$ pevně. Díky dříve zvolené konstrukci je řešení pro $\mathbf{S}$ p-měkké prahování

$$\mathbf{S}^{n+1} = \mathrm{soft}^p_{\mu\lambda}(\mathbf{M} - \mathbf{L}^n - \mathbf{Y}^n). \tag{2.47}$$

Dále zvolme $\mathbf{S}$ pevně. Potřebujeme minimalizovat $\sigma(\mathbf{L})$. Tato minimalizace může být vložena dovnitř SVD rozkladu (důkaz v [6]). Tedy

$$\mathbf{L}^{n+1} = \mathbf{U}\text{soft}_{\mu}^{p}(\Sigma)\mathbf{V}^{*}, \text{ kde } \mathbf{U}\Sigma\mathbf{V}^{*} = \mathbf{M} - \mathbf{S}^{n+1} - \mathbf{Y}^{n}. \tag{2.48}$$

Nakonec podle ADMM metody dopočítáme Lagrangeův multiplikátor

$$\mathbf{Y}^{n+1} = \mathbf{Y}^{n} + \mathbf{S}^{n+1} + \mathbf{L}^{n+1} - \mathbf{M}. \tag{2.49}$$

Algoritmus pro nekonvexní RPCA separaci vypadá následovně:

Algoritmus 2: Nekonvexní RPCA

inicializace: $\mathbf{L}_0 = \mathbf{Y}_0 = 0$, $\mu > 0$, $0 < a < 1$, $0 \leq p < 1$;
while $\|\mathbf{M} - \mathbf{L}_{k+1} - \mathbf{S}_{k+1}\|_F > \delta\|\mathbf{M}\|_F$ **do**
  $\mathbf{S}^{n+1} = \text{soft}_{\mu\lambda}^{p}(\mathbf{M} - \mathbf{L}^{n} - \mathbf{Y}^{n})$;
  $\mathbf{L}^{n+1} = \mathbf{U}\text{soft}_{\mu}^{p}(\Sigma)\mathbf{V}^{*}$, kde $\mathbf{U}\Sigma\mathbf{V}^{*} = \mathbf{M} - \mathbf{S}^{n+1} - \mathbf{Y}^{n}$;
  $\mathbf{Y}^{n+1} = \mathbf{Y}^{n} + \mathbf{S}^{n+1} + \mathbf{L}^{n+1} - \mathbf{M}$;
  $\mu = a\mu$;
end
Výsledek: $\mathbf{L}, \mathbf{S}$

Parametr δ v algoritmu 2 udává, při jaké dosažené přesnosti se algoritmus zastaví. V každé iteraci je parametr μ zmenšen na svůj násobek s konstantou $0 < a < 1$, dokud nedosáhne své dolní meze. Jak tyto parametry a dolní mez zvolit bude rozebráno v dalších kapitolách.

Nevýhodou tohoto nekonvexního algoritmu je, že jsme schopni dokázat jeho konvergenci pouze pro $p = 1$. Pro ostatní nekonvexní případy toho schopni nejsme. Empiricky ale bylo v článku [6] vyzkoušeno, že algoritmus zkonverguje pro široké spektrum vstupních dat. Pouze pro případy, kdy je $p \ll 1$, může nastat, že pokud je algoritmus blízko ke konvergenci, začne mírně oscilovat. Ale i tyto ojedinělé případy bylo v [6] možné vyřešit, pokud v p-měkkém prahování (2.44) bylo nahrazeno $|\mathbf{t}|^p$ výrazem $\left(\sqrt{\mathbf{t} + \epsilon}\right)^p$.

2.5 Metoda dynamických modů (DMD)

Metoda dynamických modů (Dynamic Mode Decomposition), kterou budeme označovat zkráceně DMD, je matematická metoda založená na Koopmanově analýze, která se používá pro analýzu a předpověď chování komplexních systémů. DMD prvně představil Peter Schmid v roce 2010 [31] jako nástroj pro analýzu proudění tekutin v hydromechanice. Díky své schopnosti poradit si i s velmi složitými systémy, které ani často nejsme schopni popsat rovnicemi, se v dnešní době používá v množství odvětví jako je např. neurověda, finanční trhy, klimatologie, inženýrství a další [25], [34], [19].

Základní myšlenkou DMD je aproximovat složitou dynamiku systému lineárním modelem a tím ze systému získat nejdůležitější mody, které určují jeho chování. DMD tyto mody a jim odpovídající frekvence získává z měření ukazujících chování tohoto systému v průběhu času. Následnou analýzou dominantních modů získáme představu o dynamice daného systému, a dokonce můžeme předvídat jeho budoucí chování. Bavíme se ale pouze o velmi blízké budoucnosti, protože v obecném případě se nepřesnost získaná DMD rozkladem s postupujícím časem exponenciálně zvětšuje.

Od svého prvního představení byla DMD metoda předmětem rozsáhlého výzkumu a vývoje. Díky tomu byla navržena nová rozšíření základní metody, která se lépe hodí pro speciální příklady systémů a dat. Některá tato rozšíření budou dále probrána detailněji. Stejně tak byla metoda zkombinována s již známými metodami jako je např. Proper Orthogonal Decomposition [23] pro zvýšení její výkonnosti a použitelnosti.

Obecně může být řečeno, že DMD metoda je důležitým nástrojem v mnoha odvětvích nejen díky své schopnosti diagnostikování a předvídání chování komplexních systémů, ale může být použita i pro řízení těchto systémů a jejich optimalizaci.

Celá tato podkapitola je založena převážně na zdrojích [19], [32], [7],[31] a [11].

2.5.1 Formulace

Metoda DMD se typicky používá při problémech, které se dají popsat nelineárním dynamickým systémem

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t, \boldsymbol{\mu}), \tag{2.50}$$

kde $\mathbf{x}(t) \in \mathbb{R}^n$ je vektor odpovídající stavu daného dynamického systému v čase t, $\boldsymbol{\mu}$ obsahuje parametry příslušného systému a funkce $\mathbf{f}$ odpovídá dynamice systému. Typickým příkladem výše zmíněného je systém obyčejných nebo parciálních diferenciálních rovnic. Taktéž může nastat situace, kdy nejsme schopni dynamiku daného systému vyjádřit, protože rovnice, které by ji popisovaly, nejsou známé. Jedná se např. o tak složité systémy jako jsou pochody v mozku nebo pohyb oceánu. Pro takový systém tedy nelze najít exaktní řešení a je nutno využít numerických metod.

Proto spojitou funkci $\mathbf{f}$ vyjadřující dynamiku diskretizujeme jak v prostoru, tak v čase, přičemž zavedeme časový úsek Δt a diskretizujeme v každém Δt. Tedy $\mathbf{x}_k = \mathbf{x}(k\Delta t)$. Získáme tak diskrétní vyjádření systému:

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k), \tag{2.51}$$

kde $\mathbf{F}$ je dynamika nyní diskrétního systému získaná diskretizací původní dynamiky $\mathbf{f}$ v časech $t_k = k\Delta t$ a ve vybraných bodech prostoru.

Poté v daných časech t_k, $k = 1, 2, \ldots, n$ provedeme n měření. V našem případě budeme získaná měření považovat přímo za stav systému. Stavový vektor $\mathbf{x}$ je m-dimenzionální a vzniká diskretizací systému na jednotlivé prostorové části – jedná se o jednotlivá místa, ve kterých je měření prováděno. Přidáme-li k systému (2.51) počáteční podmínku $\mathbf{x}(t_0) = x_0$, získáme vhodně definovanou počáteční úlohu.

Metoda DMD přistupuje k řešení tohoto problému s předpokladem, že dynamika $\mathbf{F}$ nemusí být známa. Navíc namísto užití rovnic hledá řešení pomocí aproximace tzv. *Koopmanova operátoru*.

2.5.2 Koopmanův operátor

Koopmanův operátor je velmi často používaný operátor v množství odvětví. Jedná se např. o fyziku, biologii nebo inženýrství a to převážně pro svoji schopnost modelovat a analyzovat komplexní systémy. Je jedinečný převážně tím, že operuje na celých funkcích namísto jednotlivých bodů v prostoru a čase. Tím umožňuje zachytit chování složitých systémů, které zatím nejsme schopni pochopit pomocí tradičních metod.

Označme naměřené hodnoty systému jako $\mathbf{y} = \mathbf{g}(\mathbf{x})$, přičemž $\mathbf{g}$ je zatím neznámý vektor funkcí. Dynamiku nelineárního systému $\mathbf{F}$ tímto aproximujeme na dynamiku naměřených hodnot $\mathbf{g}(\mathbf{x})$. Nyní zavedeme zobrazení $\mathbf{K}$ popisující vývoj naměřených hodnot v jednotlivých časech t_k. Tím získáme:

$$\mathbf{g}(x_{k+1}) = \mathbf{K}\mathbf{g}(x_k) = \mathbf{g}(\mathbf{F}(x_k)). \tag{2.52}$$

Z uvedeného vztahu snadno dokážeme, že $\mathbf{K}$ je lineární operátor, protože splňuje $\mathbf{K}(c_1\mathbf{g}(\mathbf{x}_k) + c_2\mathbf{g}(\mathbf{y}_k)) = c_1\mathbf{g}(\mathbf{x}_{k+1}) + c_2\mathbf{g}(\mathbf{y}_{k+1}) = c_1\mathbf{K}\mathbf{g}(\mathbf{x}_k) + c_2\mathbf{K}\mathbf{g}(\mathbf{y}_k)$ pro jakékoliv dvě konstanty c_1, c_2 a stavové hodnoty $\mathbf{x}, \mathbf{y}$. První rovnost ve výše uvedeném vztahu platí, protože operátor $\mathbf{K}$ převádí data o Δt do budoucnosti. Tzn. že konstanty c_1, c_2 budou stejné i v čase $(k+1)\Delta t$ a součet $\mathbf{g}(\mathbf{x}_k) + \mathbf{g}(\mathbf{y}_k)$ představuje součet naměřených hodnot v čase $k\Delta t$, tedy převedeme-li daný součet o Δt do budoucnosti získáme součet naměřených hodnot v čase $(k+1)\Delta t : \mathbf{g}(\mathbf{x}_{k+1}) + \mathbf{g}(\mathbf{y}_{k+1})$. Operátor $\mathbf{K}$ budeme nazývat *Koopmanův operátor*.

Výše zmíněný postup, kdy jsme stavový vektor $\mathbf{x}$ nahradili vektorem naměřených hodnot $\mathbf{g}(\mathbf{x})$, lze chápat jako vložení dynamiky stavového vektoru do ekvivalentní dynamiky naměřených hodnot. Tedy nelineární dynamika $\mathbf{F}$ v $\mathbf{x}$ byla převedena na ekvivalentní lineární dynamiku v $\mathbf{g}(\mathbf{x})$. Zároveň byl daný konečný nelineární systém převeden na nekonečný lineární systém.

Linearizaci je možné provést i jiným způsobem. Např. obsahuje-li systém nelineární polynomy, můžeme přidat i polynomické prvky a definovat $\mathbf{g}(\mathbf{x})$ jako prvek obsahující $\mathbf{x}$ a vyšší mocniny $\mathbf{x}$. Tato linearizace se nazývá *Carlemanova*. V praxi

není používaná tak často jako Koopmanova, protože často vede k divergencím – neschopnosti uzavřít daný systém.

Můžeme tedy říct, že Koopmanův operátor je lineární operátor v nekonečně rozměrném prostoru funkcí. Pokud máme diskrétní dynamický systém $x_{k+1} = \mathbf{F}(x_k)$, kde $\mathbf{x}$ je stavový vektor a $\mathbf{F}$ dynamika systému, pak Koopmanův operátor transformuje funkci popisující stav systému v čase t_k na funkci popisující stav systému v čase t_{k+1}:

$$\mathbf{K}\mathbf{g}(x_k) = \mathbf{g}(x_{k+1}). \tag{2.53}$$

Nyní můžeme přejít k využití tohoto operátoru v tzv. *Koopmanově analýze*. Jejím cílem je ze stavových dat systému najít vektor funkcí $\mathbf{g}$ (často se jedná o identitu, tedy získané hodnoty odpovídají přímo hodnotám získaným při měření daného systému $\mathbf{g}(\mathbf{x}) = \mathbf{x}$) a vyjádřit ji pomocí Koopmanových modů a vlastních čísel.

K tomu nejdříve, za předpokladu, že Koopmanův operátor je reprezentován čtvercovou regulární maticí, provedeme rozklad Koopmanova operátoru na vlastní vektory a vlastní čísla:

$$\mathbf{K}c_j(\mathbf{x}) = \lambda_j c_j(\mathbf{x}), \quad j = 1, 2, \dots \tag{2.54}$$

Dále mohou být hodnoty $\mathbf{g}$ vyjádřeny obecně pomocí Koopmanových vlastních funkcí c_j jako:

$$\mathbf{g}(\mathbf{x}) = \sum_{j=1}^{\infty} c_j(\mathbf{x})\boldsymbol{\phi}_j, \tag{2.55}$$

kde $\boldsymbol{\phi}_j$, $j = 1, 2 \dots$ jsou vektory vhodných koeficientů. Dále je budeme označovat jako *Koopmanovy mody*.

Pro další krok předpokládejme, že všechny složky $\mathbf{g}$ náleží do lineárního obalu vlastních funkcí c_j. Pak užitím předešlého vztahu (2.55) a definice Koopmanova operátu (2.52) můžeme napsat:

$$\mathbf{g}(\mathbf{x}_k) = \sum_{j=1}^{\infty} \lambda_j^k c_j(\mathbf{x}_0)\boldsymbol{\phi}_j. \tag{2.56}$$

Tedy $c_j(\mathbf{x}_0)$ je počáteční koeficient získaný z počátečního stavu a *Koopmanova vlastní čísla* λ_j určují jakou frekvenci každý Koopmanův mod má a jak se jeho velikost mění v čase.

Budeme-li předpokládat, že dynamika systému je lineární, tzn. lze napsat $\mathbf{F}(\mathbf{x}) = \mathbf{A}\mathbf{x}$, poté se dá snadno ukázat, že vlastní čísla matice $\mathbf{A}$ jsou zároveň vlastními čísly Koopmanova operátoru. A jsou-li navíc naměřené hodnoty rovny stavovým: $\mathbf{g}(\mathbf{x}) = \mathbf{x}$, pak Koopmanovy mody ϕ_j odpovídají přímo vlastním vektorům matice $\mathbf{A}$ (je ukázáno v [30]).

2.5.3 Odvození DMD

Koopmanova analýza je důležitá pro DMD, protože tento algoritmus z konečného množství naměřených dat aproximuje Koopmanovy mody a vlastní čísla, čímž poskytuje časoprostorovou dekompozici daných dat. DMD lze považovat za variaci Arnoldiho algoritmu. Jedná se o numerickou metodu používanou pro hledání aproximace vlastních čísel a vektorů matice.

Algoritmus DMD byl původně navržen s předpokladem, že data jsou naměřena v pravidelných časových intervalech. Příkladem je i náš problém separace ve videu, jelikož snímky videa jsou pořízeny se stejným časovým rozmezím. Ovšem v dnešní době již existují vylepšení základní DMD metody, která dokážou pracovat i s daty z nepravidelného měření.

Soustředíme-li se pouze na základní verzi DMD, tak data mají dva hlavní parametry:
m – počet bodů v prostoru, ve kterých probíhá měření
n – počet provedených měření

Následně z dat vytvoříme dvě matice:

$$\mathbf{X_1} = [\mathbf{x}_1\ \mathbf{x}_2 \dots \mathbf{x}_{n-1}], \quad \mathbf{X_2} = [\mathbf{x}_2\ \mathbf{x}_3 \dots \mathbf{x}_n].$$

Tedy první matice obsahuje prvních $n-1$ měření a druhá posledních $n-1$ měření. Následně chceme najít lineární operátor $\mathbf{A}_{m\times m}$ (jedná se o aproximaci Koopmanova operátoru), který převede data o Δt do budoucna

$$\mathbf{X_2} = \mathbf{A}\mathbf{X_1}. \tag{2.57}$$

Operátor $\mathbf{A}$ hledáme ve smyslu metody nejmenších čtverců $\|\mathbf{X_2} - \mathbf{A}\mathbf{X_1}\|_F$, kde $\|.\|_F$ je Frobeniova norma. Potom řešením je:

$$\mathbf{A} = \mathbf{X_2}\mathbf{X_1}^+, \tag{2.58}$$

kde $\mathbf{X_1}^+$ značí Moore–Penroseovu pseudoinverzi.

Pro další práci je dobré mít na paměti, že pro obě matice $\mathbf{X_1}, \mathbf{X_2}$ bude většinou platit $m \gg n$, neboli mají mnohem více řádků než sloupců. Toto chování lze ukázat i na našem příkladu videa, kdy počet pixelů v jednotlivých snímcích je mnohem vyšší než celkový počet snímků (omezíme-li se na videa dlouhá okolo minuty se standardním počtem snímků za sekundu). Pokud bychom v takovém případě provedli přímo $\mathbf{A} = \mathbf{X_2}\mathbf{X_1}^+$ násobíme dvě „dlouhé" obdélníkové matice a výsledkem násobení bude „obrovská" matice $m \times m$, která kompletně ignoruje fakt, že většina dynamických systémů vykazuje nízkohodnostní chování. Navíc spočítat rozklad na vlastní čísla takto velké matice bude výpočetně náročný úkol.

Další možností pro výpočet $\mathbf{A}$ je použít QR rozklad (příklad užití v [31]). Ten se ale v praxi často ukazuje jako špatně podmíněný algoritmus a navíc je často schopen

extrahovat pouze první dva dominantní mody. To platí zejména obsahují-li data šum nebo pokud obsahují množství nepřesností. Proto raději použijeme postup poprvé navržený v [31], který bude rozebrán v další podkapitole.

2.5.4 DMD algoritmus

Nejprve provedeme singulární rozklad matice $\mathbf{X_1}$

$$\mathbf{X}_1 = \mathbf{U}\Sigma\mathbf{V}^*, \tag{2.59}$$

kde $\mathbf{V}^*$ je hermitovská transpozice matice $\mathbf{V}$.

Jelikož předpokládáme, že dynamický systém obsahuje nízkohodnostní strukturu, provedeme redukci dimenze na hodnost r, kterou volíme podle daného systému. Tedy $\mathbf{U} \in \mathbb{C}^{m\times r}, \Sigma \in \mathbb{C}^{r\times r}$ a $\mathbf{V} \in \mathbb{C}^{n-1\times r}$. Tímto krokem se omezíme pouze na r dominantních složek, které řídí chování systému a ostatní nebudeme brát v úvahu.

Pak pseudoinverzní matici $\mathbf{X_1}^+$ vyjádříme pomocí SVD rozkladu (jak bylo ukázáno v podkapitole 1.5) následovně

$$\mathbf{X_1}^+ = \mathbf{V}\Sigma^{-1}\mathbf{U}^*. \tag{2.60}$$

Poté platí

$$\mathbf{A} = \mathbf{X}_2\mathbf{V}\Sigma^{-1}\mathbf{U}^*. \tag{2.61}$$

Abychom snížili výpočetní nároky, je výhodné počítat s maticí $\tilde{\mathbf{A}}$, která je $r \times r$ projekcí matice $\mathbf{A}$ na POD (Proper Orthogonal Decomposition [23]) mody. Jejich užití v mnoha aplikacích umožňuje snížit dimenzi dat a odstranit redundance. Tedy

$$\tilde{\mathbf{A}} = \mathbf{U}^*\mathbf{A}\mathbf{U} = \mathbf{U}^*\mathbf{X}_2\mathbf{V}\Sigma^{-1}. \tag{2.62}$$

Poté matice $\tilde{\mathbf{A}}$ definuje nízkohodnostní lineární model daného systému na v POD prostoru

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k. \tag{2.63}$$

Chceme-li se vrátit zpátky na model s vyšší dimenzí, stačí použít:

$$\mathbf{x}_k = \mathbf{U}\tilde{\mathbf{x}}_k. \tag{2.64}$$

Nyní spočítáme rozklad na vlastní čísla matice $\tilde{\mathbf{A}}$. Rozklad je možné provést, protože matice $\tilde{\mathbf{A}}$ je zcela jistě čtvercová a jsou-li všechna singulární čísla v matici Σ nenulová, tak je i regulární.

$$\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}, \tag{2.65}$$

kde $\mathbf{\Lambda}$ je diagonální matice vlastních čísel (budeme je nazývat *DMD vlastní čísla*) a $\mathbf{W}$ je matice příslušných vlastních vektorů. Tedy

$$\mathbf{A}\mathbf{U} = \mathbf{U}\tilde{\mathbf{A}} = \mathbf{U}\mathbf{W}\mathbf{\Lambda}\mathbf{W}^{-1}.$$

Výraz vynásobíme zprava maticí $\mathbf{W}$

$$\mathbf{AUW} = \mathbf{UW\Lambda}$$

a označíme $\mathbf{\Phi} = \mathbf{UW}$, pak

$$\mathbf{A\Phi} = \mathbf{\Phi\Lambda},$$

kde $\mathbf{\Phi}$ je matice hledaných DMD modů. Mody získané jako $\mathbf{\Phi} = \mathbf{UW}$ se nazývají *projektované DMD mody*. Tyto mody jsou dostatečné pro použití, pokud nehledáme nenulový mod patřící nulovému vlastnímu číslu $\lambda_k = 0$. Je-li cílem užití DMD zkonstruování matematického modelu chování systému, je potřeba znát všechny mody patřící k nulovému vlastnímu číslu. Tyto mody většinou vyjadřují stacionární chování systému a jejich vynecháním můžeme způsobit značné nepřesnosti v predikcích chování. V takovém případě musíme použít tzv. *exaktní DMD mody*, které získáme jako

$$\mathbf{\Phi} = \mathbf{X}_2\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}. \tag{2.66}$$

Tyto mody se nazývají exaktní, protože bylo v [13] dokázáno, že se jedná o přesné vlastní vektory matice $\mathbf{A}$. Pokud se nejedná o výše zmíněný případ nulového vlastního čísla, kdy je potřeba použít přesné mody, budeme používat projektované mody.

Pro účely předpovídání časové dynamiky systému převedeme DMD mody na jiné, které autoři [11] nazývají *Fourierovy mody*:

$$\omega_j = \frac{\ln(\lambda_j)}{\Delta t}, \tag{2.67}$$

kde λ_j jsou DMD vlastní čísla a Δt je časový krok $\Delta t = t_{j+1} - t_j$. Protože DMD vlastní čísla λ_j jsou obecně komplexní, logaritmus komplexního čísla spočítáme v polárních souřadnicích. Tedy komplexní číslo $z = x + \mathrm{i}y$ převedeme na $z = |z|\mathrm{e}^{\mathrm{i}\phi}$ a $\ln(|z|\mathrm{e}^{i\phi}) = \ln|z| + i\phi$. Pak reálná část ω_j reguluje růst a pokles modů, zatímco imaginární část ω_j určuje oscilaci těchto modů.

S využitím aproximace vlastní čísel ve formě Fourierových modů a užitím DMD modů můžeme zkonstruovat řešení v jakémkoliv čase t po získání prvního vektoru $\mathbf{x}_1$ v čase $t_1 = 0$. Tedy pro $t > t_1 = 0$ platí

$$\mathbf{X}_{\mathrm{DMD}}(t) = \sum_{j=1}^{r} b_j \boldsymbol{\varphi}_j \mathrm{e}^{\omega_j t} = \mathbf{\Phi}\mathrm{e}^{\mathbf{\Omega}t}\mathbf{b}, \tag{2.68}$$

kde $\mathbf{\Phi}$ jsou DMD mody, $\mathbf{\Omega}$ je diagonální matice na jejíž diagonále jsou Fourierovy mody ω_j a t je čas, ve kterém nás zajímá chování dat. Jediný neznámý výraz, který zde vystupuje, je vektor počátečních koeficientů $\mathbf{b}$. Ten je možné spočítat z počátečního měření $\mathbf{x}_1$ v čase $t_1 = 0$. Tedy $\mathbf{x}_1 = \mathbf{\Phi b}$, protože z exponenciálního členu se stala jednotková matice. Jelikož je matice $\mathbf{\Phi}$ v obecném případě obdélníková, použijeme opět pseudoinverzi:

$$\mathbf{b} = \mathbf{\Phi}^{+}\mathbf{x_1}. \tag{2.69}$$

Výsledný algoritmus je tedy následovný.

Algoritmus 3: Dynamic Mode Decomposition (DMD)

1. SVD rozklad: $\mathbf{X}_1 = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$;
2. Výběr r a následná redukce dimenze na $\mathbf{U} \in \mathbb{C}^{m\times r}, \mathbf{\Sigma} \in \mathbb{C}^{r\times r}, \mathbf{V} \in \mathbb{C}^{n-1\times r}$;
3. Projekce $\mathbf{A}$ na $\tilde{\mathbf{A}}$: $\tilde{\mathbf{A}} = \mathbf{U}^*\mathbf{X}_2\mathbf{V}\mathbf{\Sigma}^{-1}$;
4. Výpočet vlastních čísel $\tilde{\mathbf{A}}$: $\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}$;
5. Určení DMD modů: $\mathbf{\Phi} = \mathbf{X}_2\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}$;
6. Výpočet Fourierových modů: $\mathbf{\Omega} = \frac{\ln \mathbf{\Lambda}}{\Delta t}$;
7. Získání počátečních koeficientů $\mathbf{b}$: $\mathbf{b} = \mathbf{\Phi}^+\mathbf{x_1}$;
8. Dosazení do řešení: $\mathbf{X}_{\mathrm{DMD}}(t) = \sum_{j=1}^{r} b_j \boldsymbol{\varphi}_j \mathrm{e}^{\omega_j t}$;

Tento algoritmus je ovšem pouze základní variantou DMD. Protože poptávka po zpracování dat z velkých systémů je čím dál větší, vyvíjejí se vylepšení tohoto základního algoritmu. Tato vylepšení dosahují pro speciální případy systémů lepších výsledků.

Máme-li např. model, o kterém víme, že je silně nelineární, nemusí být lineární aproximace pomocí DMD dostatečná. V takovém případě je lepší použít tzv. *rozšířené DMD* (extended DMD). Potřebujeme-li co nejpřesnější výsledky při analýze pouze malého množství měření, je vhodné použít tzv. *přesné DMD* (exact DMD). Tato metoda se používá převážně, když víme, že naše data neobsahují téměř žádný šum. DMD je možné použít i pro optimální řízení systému. V takovém případě existuje tzv. *DMD s řízením* (DMD with control). Výše zmíněné metody jsou podrobněji rozebrány v [32].

V neposlední řadě existuje tzv. *Multiresolution DMD*, které může být použito např. pro rozklad videa na více složek podle jejich dynamiky. Tato metoda bude podrobněji rozebrána v podkapitole 2.6.

2.5.5 DMD pro separaci videa

Jak už bylo dříve zmíněno, data ve formě videa jsou vhodná k užití DMD, protože jednotlivé snímky jsou rovnoměrně rozložené v čase a pixely každého snímku představují prostorové body, ve kterých proběhlo měření systému. Každý snímek je vektorizován na sloupec, který odpovídá jednomu měření. Tím je zajištěno, že každé měření má stejný počet bodů, ve kterých je provedeno, protože tyto body odpovídají jednotlivým pixelům každého snímku. Navíc jsou snímky ve videu pořízeny vždy za stejný časový úsek Δt, což je ideální případ pro užití DMD metody.

Chceme-li zrekonstruovat celé video, zvolme prvních $n-1$ snímků jako $\mathbf{X}_1$ a posledních $n-1$ snímků jako $\mathbf{X}_2$. Nyní provedeme DMD a spočteme DMD mody

a Fourierovy mody. Vektor počátečních koeficientů $\mathbf{b}$ získáme z $\mathbf{b} = \mathbf{\Phi}^{+}\mathbf{x_1}$, kde $\mathbf{x_1}$ je první snímek videa.

Uvažujeme-li vektor časů $\mathbf{t} = [t_1, t_2, \ldots, t_n]$, ve kterých byly jednotlivé snímky pořízeny, pak matice $\mathbf{X}_{\mathrm{DMD}}$ představující celé video může být zrekonstruována následovně:

$$\mathbf{X}_{\mathrm{DMD}} = \sum_{j=1}^{r} b_j \boldsymbol{\varphi}_j \mathrm{e}^{\omega_j \mathbf{t}} = \mathbf{\Phi} \mathrm{e}^{\mathbf{\Omega} t} \mathbf{b}, \tag{2.70}$$

kde výraz $\mathrm{e}^{\mathbf{\Omega} t}$ představuje diagonální matici, na jejíž diagonále leží členy $\mathrm{e}^{\omega_j \mathbf{t}}$. Tedy získaná matice $\mathbf{x}_{\mathrm{DMD}}$ by měla odpovídat matici získané z původních vstupních dat.

Jelikož vektor koeficientů $\mathbf{b}$ je počítán z prvního snímku, můžeme říct, že matice Fourierových modů $\mathbf{\Omega}$ určuje, jak se složky v prvním snímku videa v čase mění. S tímto poznatkem lze říct, že každá složka videa, která se v čase nemění nebo se mění jen velmi málo, má odpovídající Fourierův mod umístěn v komplexním prostoru blízko počátku $|\omega_j| \approx 0$.

Díky tomuto zjištění je možné odseparovat pozadí od dynamické složky. Předpokládejme, že pro ω_p, kde $p \in P \subset R = \{1, 2 \ldots, r\}$, platí $|\omega_p| \approx 0$ a že výraz $|\omega_j|$ není blízký nule $\forall j \notin P$, potom

$$\mathbf{X}_{\mathrm{DMD}} = \underbrace{\sum_{p \in P} b_p \boldsymbol{\varphi}_p \mathrm{e}^{\omega_p \mathbf{t}}}_{\text{pozadí}} + \underbrace{\sum_{j \notin P} b_j \boldsymbol{\varphi}_j \mathrm{e}^{\omega_j \mathbf{t}}}_{\text{dynamická složka}} . \tag{2.71}$$

Pokud $\mathbf{X} \in \mathbb{R}^{m \times n}$, pak i $\mathbf{X}_{\mathrm{DMD}} \in \mathbb{R}^{m \times n}$. Nicméně jednotlivé složky $b_j \boldsymbol{\varphi}_j e^{\omega_j \mathbf{t}}$ jsou komplexní, tzn. reálnou matici dostaneme až sečtením všech složek. Tento fakt působí problém při separaci, protože očekáváme reálné hodnoty a pro přesnost výsledků musíme vědět, jak naložit s těmito komplexními prvky:

Předpokládáme, že nízkohodnostní složka vypadá následovně:

$$\mathbf{L} = \sum_{p} b_p \boldsymbol{\varphi}_p \mathrm{e}^{\omega_p \mathbf{t}}.$$

Využijeme vlastnosti $\mathbf{X}_{\mathrm{DMD}} = \mathbf{L} + \mathbf{S}$, kde $\mathbf{S}$ představuje dynamickou (řídkou) složku videa a použitím následujícího vzorce zajistíme, že řídká složka bude mít reálné hodnoty.

$$\mathbf{S} = \mathbf{X}_{\mathrm{DMD}} - |\mathbf{L}|,$$

kde $|\cdot|$ značí absolutní hodnotu z každého prvku matice.

Tento postup má jeden problém. Může se stát, že pixely matice $\mathbf{S}$ budou mít negativní hodnotu. Možným řešením je tyto hodnoty dát do pomocné matice $\mathbf{R}$ a tu pak přičíst ke složce $\mathbf{L}$ a odečíst ji od složky $\mathbf{S}$. Další možností je tento problém vyřešit až při vykreslení videa a neovlivňovat získané matice, aby nebyla jejich nepřesnost ještě zvýšena.

2.5.6 Srovnání s RPCA

Obě metody jsou, jak bude dále ukázáno, schopné velmi dobře odseparovat pozadí od dynamické složky. V případě DMD metody záleží dobrý výsledek především na volbě parametru r, který redukuje dimenzi modelu. Dále také na prahu ε, který určuje, které mody leží dostatečně blízko počátku $|\omega_j| \leq \varepsilon$. Jedná se o mody odpovídají pozadí.

Existují typy videí, se kterými si DMD na rozdíl od RPCA neumí dostatečně dobře poradit. Příkladem takového chování je video, kde objekty náhle na určitou dobu zastavují a pak se opět náhle pohybují. Abychom toto chování mohli popsat, bude potřeba velké množství Fourierových modů. Ovšem tyto mody nemusí být k dispozici v důsledku volby malého parametru r. I když k dispozici jsou, je pravděpodobné, že tyto mody budou zachycovat už i pomalu se pohybující objekty a tím přinesou množství nepřesností do výsledného pozadí.

Naopak obrovskou výhodou DMD nad RPCA je jeho poměrně nízká výpočetní náročnost. V případě obou technik je nejnáročnějším krokem výpočet SVD rozkladu. Tento krok ovšem v DMD provedeme pouze jednou, zatímco pro RPCA je prováděn v každé iteraci. Tedy DMD algoritmus může být při efektivní implementaci použit i na separaci videí v reálném čase. Jelikož množství průmyslových kamer má rychlost snímání pouze okolo 30 snímků za sekundu, DMD je schopno tyto úseky spočítat do 0,1 s.

2.6 Multiresolution DMD

Tato podkapitola vychází ze zdrojů [20],[21] a [19].

Nejprve jako motivaci k zavedení rozšíření standardního DMD uveďme příklad. Tím může být video zachycující dopravní křižovatku se světelným značením. Když svítí červená, auta stojí – patří do pozadí. Při rozsvícení zelené se náhle rozjedou a již spadají do dynamické složky. Standardní DMD předpokládá, že systém nebude v průběhu celého času, kdy je zkoumán, vykazovat takto nekonzistentní chování. Proto provedeme-li separaci pomocí DMD, pravděpodobně zůstanou na pozadí poloprůhledně viditelná stojící auta, i když tam v danou chvíli už nebudou a to způsobí velké množství nepřesností. Z tohoto důvodu je vhodné místo standardního DMD použít jeho vylepšení tzv. *multiresolution DMD*, které bude dále označováno jako mrDMD.

Metoda mrDMD je užitečná nejen při analýze systémů, jejichž chování se v jednotlivých časových segmentech mění. Přičemž je důležité zdůraznit, že se jedná pouze o rozdílné chování v čase nikoliv v prostoru. Vhodná je také pro vysokorozměrné systémy, protože ve srovnání s DMD je možné snížit výpočetní náročnost. Dále se

hodí pro data obsahující šum nebo pro neúplná data, protože je dekompozicí dat do více segmentů schopna získat dominantní mody i takto nepřesných dat.

Algoritmus mrDMD vzniká spojením standardního DMD s vlnkovou transformací, proto ji nejprve krátce představíme.

2.6.1 Vlnková transformace

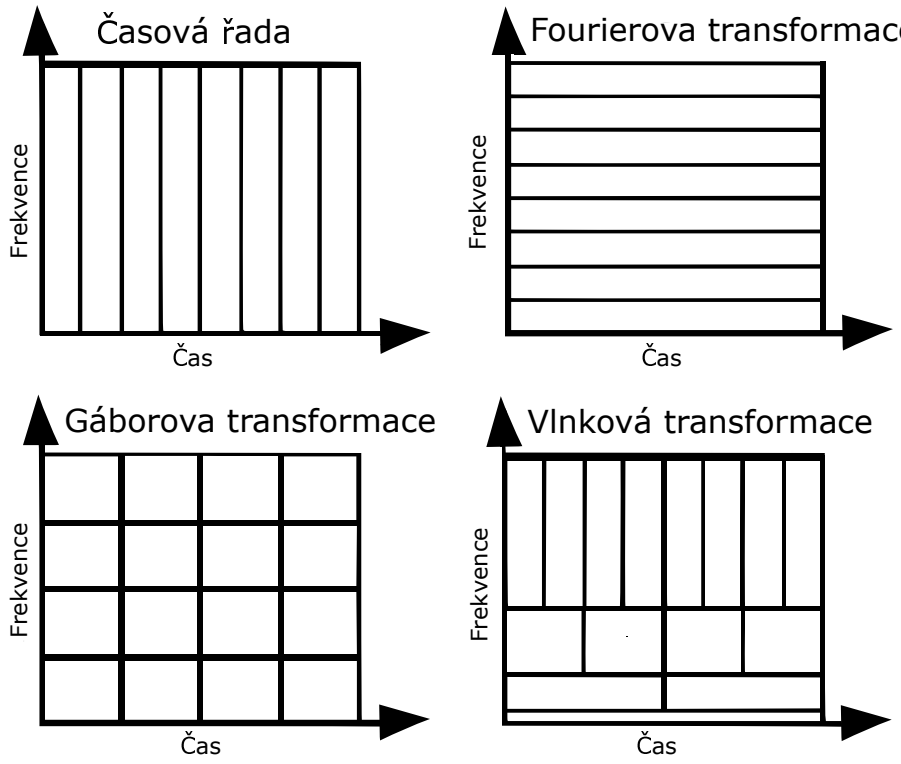
Tato podkapitola pouze naznačuje princip vlnkové transformace. Detailnější informace ohledně této metody jsou dostupné např. ve článku [39].

Jednou z nejdůležitějších metod pro analýzu časových řad je *Fourierova transformace*. Využívá se zejména v oblasti zpracování zvukových a obrazových signálů, kde umožňuje převod mezi časovou a frekvenční doménou. Fourierova transformace má bohužel i značná omezení. Transformací časového signálu je získáno celé frekvenční spektrum daného signálu, ale zároveň je ztracena veškerá informace o časové doméně.

Z tohoto důvodu vznikla snaha překonat omezení Fourierovy transformace a najít metodu, která by byla schopna lokalizovat signál zároveň v časové i frekvenční doméně. Toho bylo dosaženo představením tzv. *Gaborova jádra*, což je funkce sloužící k vytvoření časových „oken“, která jsou následně aplikována na signál. Tedy v daném časovém okně je extrahováno frekvenční spektrum signálu a okno je posunuto o kus dál. Daný postup se nazývá *Gaborova transformace* a její pomocí je získáno nejen celé přibližné frekvenční spektrum, ale také časové úseky, ve kterých získané frekvence proběhly.

Bohužel i tato metoda má svá omezení. Jakákoliv složka signálu s vlnovou délkou větší než je délka časového okna, bude kompletně vynechána. Tomuto problému je možné předejít zvolením delšího okna, ale tím je značně sníženo získané časové rozlišení signálu. Tedy čím více frekvenčního spektra jsme schopni získat, tím méně získáme informací o časové doméně a naopak.

Využitím základního principu Gaborovy transformace – posun časového okna a přidáním nového principu – škálování časového okna je získána základní myšlenka tzv. *vlnkové transformace*. Její princip je následující: Nejdříve jsou extrahovány nízké frekvence při nízkém časovém rozlišení. Nízké frekvence obvykle mají dlouhou vlnovou délku, tedy dějí se na delším časovém úseku, a proto pro jejich analýzu není potřebné vysoké časové rozlišení. Následně je časové okno zmenšeno, čímž jsou získány vyšší frekvence už při lepším časovém rozlišení. Takto je pokračováno až do požadované přesnosti časového rozlišení, kdy jsou extrahovány nejvyšší frekvence. Daným postupem získáme nejen celé frekvenční spektrum, ale i poměrně přesné informace, v jakých časových úsecích tyto frekvence nastaly.



Obr. 2.1: Schematické porovnání Fourierovy, Gaborovy a vlnkové transformace.

Na obrázku 2.1 je schematicky zobrazen rozdíl mezi časovou řadou a její Fourierovou, Gaborovou a vlnkovou transformací. Máme-li časovou řadu, tak známe pouze časovou doménu signálu, ale nevíme nic o frekvencích signálu. Užitím Fourierovy transformace ztratíme veškerou informaci o časové doméně, ale získáme celé frekvenční spektrum signálu. Provedením Gaborovy transformace získáme část frekvenčního spektra i s částí časové domény. Ovšem využitím vlnkové transformace získáme v ideálním případě nejen cele frekvenční spektrum, ale i celé příslušné časové rozlišení.

2.6.2 Formulace mrDMD

Jak bylo ukázáno v předešlé kapitole, pomocí DMD získáme mody reprezentující dynamiku systému. V aplikacích, jako je separace pozadí ve videu, jsou vybrány pouze pomalé mody, které odpovídají pozadí. Metoda mrDMD využívá této vlastnosti ve spojení s vlnkovou transformací. Ve standardním DMD je potřeba vstupní data, resp. počet snímků n zvolit tak, aby mohla být získána aproximace celé dynamiky systému – aby byly přítomny všechny vysokofrekvenční i nízkofrekvenční složky. V mrDMD je potřeba vstupní data – počet snímků n zvolit tak, aby bylo možné získat mody s nejnižší frekvencí. Následně jsou tyto mody odstraněny a původní data rozdělena na poloviny – každá obsahuje $n/2$ snímků. Toto rozdělení lze

považovat za zvolení menšího časového okna. Poté opět provedeme DMD na každé polovině zvlášť a vybereme mody s nejnižší frekvencí. Daný postup je opakován, dokud není dosaženo požadované přesnosti separace.

Výše zmíněný postup je možné zapsat matematicky následovně: Provedeme rekonstrukci původního videa pomocí DMD:

$$\mathbf{X}_{\mathrm{DMD}} = \sum_{k=1}^{n_1} b_k \boldsymbol{\varphi}_k \mathrm{e}^{\omega_k \mathbf{t}} + \sum_{k=n_1+1}^{n} b_k \boldsymbol{\varphi}_k \mathrm{e}^{\omega_k \mathbf{t}}, \tag{2.72}$$

kde $\boldsymbol{\varphi}_k$ reprezentuje v první sumě pomalé mody a ve druhé sumě všechny zbylé rychlejší mody. Sumu s pomalými mody můžeme dát stranou (reprezentuje pozadí videa) a soustředíme se pouze na druhou sumu, kterou označíme $\mathbf{X}_n(\mathrm{fast})$ a rozdělíme ji na poloviny:

$$\mathbf{X}_n(\mathrm{fast}) = \sum_{k=n_1+1}^{n} b_k \boldsymbol{\varphi}_k \mathrm{e}^{\omega_k \mathbf{t}} = \mathbf{X}^{(1)}_{n/2} + \mathbf{X}^{(2)}_{n/2}, \tag{2.73}$$

kde matice $\mathbf{X}^{(1)}_{n/2}$ obsahuje prvních $n/2$ sloupců a $\mathbf{X}^{(2)}_{n/2}$ obsahuje zbylých $n/2$ sloupců původní matice $\mathbf{X}_n$. Nyní na každé z těchto matic provedeme DMD a odseparujeme pomalé mody. Označíme-li $\boldsymbol{\varphi}^{(1)}_k$ pomalé mody získané pomocí DMD z matice celého videa, pak pomalé mody odseparované na této úrovni jsou $\boldsymbol{\varphi}^{(2)}_k$.

Metoda mrDMD tedy rekurzivně odstraňuje nízkofrekvenční komponenty (určené pomalými mody) a tvoří nové matice $\mathbf{X}_{n/2}, \mathbf{X}_{n/4}, \mathbf{X}_{n/8}, \ldots$ až dokud není dosaženo požadovaného multirezolučního rozkladu. Poté mrDMD rekonstrukce videa vypadá následovně:

$$\mathbf{X}_{\mathrm{mrDMD}} = \sum_{k=1}^{n_1} b^{(1)}_k \boldsymbol{\varphi}^{(1)}_k \mathrm{e}^{\omega^{(1)}_k \mathbf{t}} + \sum_{k=1}^{n_2} b^{(2)}_k \boldsymbol{\varphi}^{(2)}_k \mathrm{e}^{\omega^{(2)}_k \mathbf{t}} + \sum_{k=1}^{n_3} b^{(3)}_k \boldsymbol{\varphi}^{(3)}_k \mathrm{e}^{\omega^{(3)}_k \mathbf{t}} + \ldots, \tag{2.74}$$

kde $\boldsymbol{\varphi}^{(\ell)}_k$, $\omega^{(\ell)}_k$ jsou DMD mody a Fourierovy mody získané separací na ℓ-té úrovni, $b^{(\ell)}_k$ jsou počáteční vektory získané z prvního snímku daného intervalu a n_ℓ je počet pomalých modů získaných na úrovni ℓ.

Pro formální zápis mrDMD je nutné zavést následující indexy:

$$\ell = 1, 2, \ldots, L\text{: počet dekompozičních úrovní,} \tag{2.75}$$

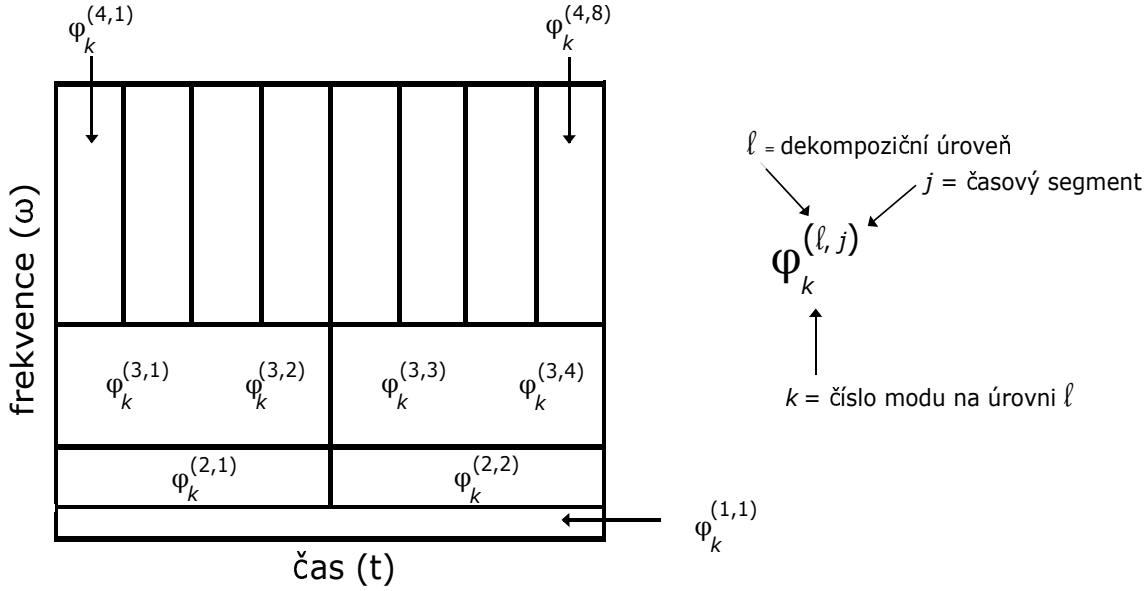
$$j = 1, 2, \ldots, J\text{: počet časových segmentů na úrovni } \ell\text{, platí } J = 2^{\ell-1}, \tag{2.76}$$

$$k = 1, 2, \ldots, n_\ell\text{: počet modů extrahovaných na úrovni } \ell. \tag{2.77}$$

Dále zavedeme charakteristickou funkci

$$f^{\ell,j}(t) = \begin{cases} 1 & t \in [t_j, t_{j+1}] \\ 0 & \text{jinde,} \end{cases} \tag{2.78}$$

která má nenulové hodnoty pouze v intervalu odpovídajícímu časovému segmentu j.



Obr. 2.2: Ilustrace dekompozičních úrovní vzniklých metodou mrDMD. $\boldsymbol{\varphi}_k^{(\ell,j)}$ značí pomalé mody získané na ℓ-té dekompoziční úrovni v j-tý časový segment. Poté trojice ℓ, j, k jednoznačně určuje daný mod získaný z dekompozice.

S využitím výše uvedeného lze mrDMD řešení formálně zapsat jako:

$$\mathbf{X}_{\mathrm{mrDMD}} = \sum_{\ell=1}^{L} \sum_{j=1}^{J} \sum_{k=1}^{n_\ell} f^{\ell,j}(t) b_k^{(\ell,j)} \boldsymbol{\varphi}_k^{(\ell,j)} \mathrm{e}^{\omega_k^{(\ell,j)} \mathbf{t}}. \tag{2.79}$$

Tato definice mrDMD obsahuje informace o počtu modů, které extrahujeme na dané úrovni a v daném časovém segmentu. Na obr. 2.2 je řešení (2.79) vizualizováno.

Z obrázku 2.2 je i na první pohled patrná podobnost mrDMD metody s vlnkovou transformací. Charakteristická funkce $f^{\ell,j}(t)$ se zde chová jako filtr pro získání jednotlivých časových segmentů. Tedy chování $f^{\ell,j}(t)$ lze přirovnat k posouvání a smršťování časového okna při užití Gaborovy transformace.

Řešení mrDMD (2.79) může být také chápáno jako řešení diskrétního lineárního systému $\mathbf{x}_{t+1}^{(\ell,j)} = \mathbf{A}^{(\ell,j)} \mathbf{x}_t^{(\ell,j)}$ ve smyslu metody nejmenších čtverců pro každý časový segment j a dekompoziční úroveň ℓ. Tohoto poznatku je využito v následujícím algoritmu tím, že částečně využívá standardní metodu DMD.

Algoritmus 4: Multi Resolution Dynamic Mode Decomposition (mrDMD)

1. Výběr daného okna pro dekompozici na úrovni ℓ v časovém segmentu j. Sestrojení matic $\mathbf{X}_1$, $\mathbf{X}_2$ v tomto okně.;
2. SVD rozklad: $\mathbf{X}_1 = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$;
3. Výpočet $\tilde{\mathbf{A}}^{(\ell,j)}$ projekce $\mathbf{A}^{(\ell,j)}$ na POD mody: $\mathbf{A}^{(\ell,j)} = \mathbf{X}_2\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^* \Rightarrow$ $\tilde{\mathbf{A}}^{(\ell,j)} = \mathbf{U}^*\mathbf{A}^{(\ell,j)}\mathbf{U} = \mathbf{U}^*\mathbf{X}_2\mathbf{V}\mathbf{\Sigma}^{-1}$;
4. Rozklad na vlastní čísla $\tilde{\mathbf{A}}^{(\ell,j)}$: $\tilde{\mathbf{A}}^{(\ell,j)}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}$;
5. Výběr pomalých DMD modů na dané úrovni dekompozice pomocí $\|\omega_j\| < \epsilon$;
6. Určení DMD exaktních modů: $\mathbf{\Phi}^{(\ell,j)} = \mathbf{X}_2\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}$;
7. Rekonstrukce matice za použití zbylých modů: $\mathbf{X}_n(\text{fast}) = \sum_{k=n_1+1}^{n} b_k\boldsymbol{\varphi}_k \mathrm{e}^{\omega_k \mathbf{t}}$;
8. Rozdělení časového okna na poloviny a zopakování kroků 1–7 pro každou polovinu daného okna na úrovni $\ell + 1$;

Výše zmíněný algoritmus obsahuje dva parametry. První je zřejmý – parametr ϵ určující, kdy je DMD mod v určitém časovém okně brán jako pomalý. Parametr ϵ nemusí být jen konstantní, může být např. zvolen jako funkce závisející na úrovni dekompozice, ve které se dané okno nachází.

Druhý parametr je schován v kroku 7. Je nutné vhodně vybrat první časové okno, na kterém bude celý algoritmus proveden. Bude-li příliš malé, mohou být do pozadí přidány i složky, které se pohybují. Bude-li příliš velké, nebudeme schopni prvních ℓ úrovní získat jakékoliv pomalé mody kromě prvních, které odpovídají pozadí. Možné volby těchto parametrů budou podrobněji rozebrány v dalších kapitolách.

Na závěr ukažme, proč zrovna půlení intervalu pomáhá metodě mrDMD odseparovat kromě pozadí i pohyblivé složky s podobnou rychlostí pohybu. Myšlenkou půlení intervalu je získat dostatečně malý interval tak, aby v něm dynamická složka vypadala staticky. Např. na kameře, která snímá ulici, natočíme chodce a projíždějící auta. Vezmeme-li např. pouze pět snímků, chodec se za ně téměř nestihne posunout. Proto na intervalu těchto pěti snímků vypadá jako statická složka (na rozdíl od auta, které urazí větší kus cesty). Dalším možným příkladem jsou auta stojící na světelné křižovatce, která se po rozsvícení zelené náhle rozjedou. Zvolíme-li časové okno vhodně, zachytíme stání aut v první polovině, kde představují statickou složku. Celý jejich pohyb poté zachytíme v druhé polovině, kde již představují dynamickou složku.

Bohužel půlení intervalu má také svoji nevýhodu. Na rozdíl od vlnkové transformace, kde se jednotlivé časové intervaly částečně překrývají, v metodě mrDMD jsou tyto intervaly disjunktní. Z tohoto důvodu mohou v získaných separacích nastávat

nespojité skoky. Např. je-li v první polovině intervalu jako pomalý mod vybrán také mod ϕ_j, ale tento mod ϕ_j není v druhé polovině vybrán jako pomalý, vzniká na přechodu těchto intervalů nespojitost. Pokud tento mod např. představoval pohyb chodce, bude tento chodec v první polovině intervalu součástí pozadí, ale v druhé polovině daného intervalu, přejde do např. první (dle rychlosti) pohyblivé složky. Tzn. že se v této složce na přechodu intervalů „náhle zjeví“ a zároveň náhle zmizí z pozadí, což pro člověka pozorujícího dané odseparované složky videa může působit jako nesmyslná změna.

3 Metody separace pro barevná vstupní data

V této kapitole jsou metody separace z kapitoly 2 rozšířeny pro případ, kdy jsou vstupní data barevná. Za základní tvar vstupních dat je považována série n barevných RGB snímků.

3.1 Separace po jednotlivých barevných složkách

Zřejmě nejjednodušším způsobem, jak pracovat s barevnými snímky, je provést zvolenou metodu separace zvlášť pro každou barevnou složku.

Jak již bylo ukázáno, ze vstupních dat jsou vytvořeny matice $\mathbf{M}^{\mathrm{R}}$, $\mathbf{M}^{\mathrm{G}}$ a $\mathbf{M}^{\mathrm{B}}$, kde jednotlivé sloupce $\mathbf{M}^{\mathrm{R}}$ odpovídají vektorům červené složky jednotlivých snímků (obdobně pro $\mathbf{M}^{\mathrm{G}}$ a $\mathbf{M}^{\mathrm{B}}$). Poté můžeme postupně aplikovat vybranou metodu separace na matice $\mathbf{M}^{\mathrm{R}}$, $\mathbf{M}^{\mathrm{G}}$ a $\mathbf{M}^{\mathrm{B}}$. Z první separace je získána nízkohodnostní matice $\mathbf{L}^{\mathrm{R}}$ a řídká matice $\mathbf{S}^{\mathrm{R}}$ pro červenou složku. Stejný postup je poté aplikován na zelenou i modrou složku. Výsledná barevná nízkohodností matice $\mathbf{L}$ je získána složením $\mathbf{L}^{\mathrm{R}}$, $\mathbf{L}^{\mathrm{G}}$ a $\mathbf{L}^{\mathrm{B}}$. Obdobně výsledná barevná řídká matice $\mathbf{S}$ je získána složením $\mathbf{S}^{\mathrm{R}}$, $\mathbf{S}^{\mathrm{G}}$ a $\mathbf{S}^{\mathrm{B}}$.

Zásadní výhodou tohoto postupu je, že může být aplikován na všechny metody separace ukázané v kapitole 2. Dále se nemusíme omezovat pouze na barevný prostor RGB. Snímky je možné převést do jiného barevného modelu. Poté je separace opět postupně provedena pro jednotlivé složky daného barevného modelu a výsledné matice jsou získány složením jednotlivých barevných složek.

Podstatnou nevýhodou ale je, že zcela zanedbáváme provázanost jednotlivých RGB složek. Ty se ovšem nemění nezávisle na sobě – pohybuje-li se fialový míč po scéně, pohybuje se červená i modrá složka současně.

Jedním možným přístupem, jak zachovat provázanost barevných složek, je převést všechny snímky do $\mathrm{YC_BC_R}$ barevného modelu, který na rozdíl od RGB zachovává vztah mezi jednotlivými barevnými složkami.

3.1.1 Barevný model $\mathrm{YC_BC_R}$

Tato podkapitola čerpá převážně z textu [47]. V $\mathrm{YC_BC_R}$ barevném modelu reprezentuje Y jas daného snímku, tedy komponent Y se dá považovat za daný snímek ve stupních šedi. Komponenty $\mathrm{C_B}$ a $\mathrm{C_R}$ potom reprezentují chrominanci neboli obsahují barevnou informaci daného obrazu. Přesněji komponent $\mathrm{C_B}$ vyjadřuje rozdíl mezi modrou složkou snímku a komponentem jasu Y. Tedy $\mathrm{C_B}$ ukazuje, kolik modré barvy se vyskytuje v obraze určeném jasem Y. Stejně tak komponent $\mathrm{C_R}$ vyjadřuje rozdíl mezi červenou složkou snímku a komponentem jasu Y.

Zelená složka v tomto barevném modelu není vyjádřena samostatným komponentem, protože je možné ji nepřímo dopočítat z ostatních komponent. Dopočet zelené složky je možný, protože komponent Y představuje váženou sumu RGB složek. Přesněji komponent Y je možné spočítat jako $0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B$ [47].

Model $\mathrm{YC_BC_R}$ se často užívá k reprezentaci digitálních fotografií a videí, protože dovoluje zkomprimovat data při zachování vizuální kvality obrazu. Tato komprimace je založena na faktu, že lidské oko je velmi citlivé na změny jasu a méně citlivé na změny barvy. Ponecháme-li komponent Y v co nejlepší kvalitě a zkomprimujeme-li chrominanční komponenty $\mathrm{C_B}$, $\mathrm{C_R}$, získáme zkomprimovaný snímek bez výrazného vlivu na subjektivní kvalitu obrazu.

3.2 Kvaternionové vyjádření RGB obrazu

Dalším možným přístupem, jak mezi sebou provázat jednotlivé barevné složky, je reprezentovat RGB snímky pomocí kvaternionů. K tomu je potřeba nejdříve ukázat, jak fungují základní operace v prostoru kvaternionů $\mathbb{H}$.

3.2.1 Kvaternionová algebra

Následující podkapitola vychází převážně z článků [40],[14]. Kvaternion je číslo $q = w + x\mathrm{i} + y\mathrm{j} + z\mathrm{k}$, kde $w, x, y, z \in \mathbb{R}$ a $\mathrm{i}, \mathrm{j}, \mathrm{k}$ jsou imaginární jednotky, pro které platí následující vztahy

$$\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1, \tag{3.1}$$

$$\mathrm{ij} = -\mathrm{ji} = \mathrm{k}, \tag{3.2}$$

$$\mathrm{jk} = -\mathrm{kj} = \mathrm{i}, \tag{3.3}$$

$$\mathrm{ki} = -\mathrm{ik} = \mathrm{j}. \tag{3.4}$$

K reprezentaci RGB pixelů budou použity tzv. *ryzí kvaterniony* definované následovně.

Definice 3.1. *Kvaternion q nazveme ryzí kvaternion právě tehdy, když má nulovou reálnou složku. Tedy $q = x\mathrm{i} + y\mathrm{j} + z\mathrm{k}$, kde $x, y, z \in \mathbb{R}$.*

Operace sčítání a odčítání kvaternionů je definována jako

$$\begin{aligned} q_0 \pm q_1 &= (w_0 + x_0\mathrm{i} + y_0\mathrm{j} + z_0\mathrm{k}) \pm (w_1 + x_1\mathrm{i} + y_1\mathrm{j} + z_1\mathrm{k}) \\ &= (w_0 \pm w_1) + (x_0 \pm x_1)\mathrm{i} + (y_0 \pm y_1)\mathrm{j} + (z_0 \pm z_1)\mathrm{k}. \end{aligned}$$

Dále definujeme operaci násobení kvaternionů

$$\begin{aligned} q_0 \cdot q_1 &= (w_0 + x_0\mathrm{i} + y_0\mathrm{j} + z_0\mathrm{k}) \cdot (w_1 + x_1\mathrm{i} + y_1\mathrm{j} + z_1\mathrm{k}) \\ &= (w_0w_1 - x_0x_1 - y_0y_1 - z_0z_1) + \\ &(w_0x_1 + x_0w_1 + y_0z_1 - z_0y_1)\mathrm{i} + \\ &(w_0y_1 - x_0z_1 + y_0w_1 + z_0x_1)\mathrm{j} + \\ &(w_0z_1 + x_0y_1 - y_0x_1 + z_0w_1)\mathrm{k}. \end{aligned}$$

Při bližším zkoumání toho zápisu si můžeme všimnout, že násobení kvaternionů není komutativní operace, tedy $q_0 \cdot q_1 \neq q_1 \cdot q_0$.

Definice 3.2. *Absolutní hodnota kvaternionu* $q = w + x\mathrm{i} + y\mathrm{j} + z\mathrm{k}$ *je definovaná jako*

$$|q| = \sqrt{w^2 + x^2 + y^2 + z^2}. \tag{3.5}$$

Absolutní hodnota kvaternionu odpovídá ℓ_2 normě daného kvaternionu. V dalších metodách bude potřeba i ℓ_1 norma pro kvaterniony.

Definice 3.3. ℓ_1 *norma pro kvaternion* $q = w + x\mathrm{i} + y\mathrm{j} + z\mathrm{k}$ *je definovaná jako*

$$\|q\|_1 = |w| + |x| + |y| + |z|. \tag{3.6}$$

Komplexní reprezentace kvaternionů

Kvaterniony s operací sčítání nebo i s operací násobení tvoří grupu (snadno lze ověřit, že splňují všechny tři grupové axiomy [43]). Na grupách [43] lze definovat grupový homomorfismus ρ (jedná se o zobrazení zachovávající grupovou strukturu)

$$\rho : \ G \to GL(n, F), \tag{3.7}$$

kde $GL(n, F)$ značí obecnou n-rozměrnou lineární grupu nad polem F.

Tento homomorfismus ρ je možné využít k reálné a komplexní reprezentaci kvaternionů a kvaternionových matic. Může být také využit k reálné reprezentaci komplexních čísel.

Tedy $G = (\mathbb{C}, \cdot)$ a $n = 2$, $F = \mathbb{R}$:

$$a + bi \to \begin{pmatrix} a & -b \\ b & a \end{pmatrix}. \tag{3.8}$$

Homomorfismus (3.8) není jedinou možnou reálnou reprezentací komplexního čísla. Může být využita i jakákoliv podobná matice k $\mathbf{M} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix}$. Tedy reprezentací

může být matice $\mathbf{N} = \mathbf{U}\mathbf{M}\mathbf{U}^{-1}$, kde $\mathbf{U}$ je regulární matice. Např. zvolíme-li $\mathbf{U} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, tak $\mathbf{M} = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}$.

Zvolíme-li $G = (\mathbb{H}, \cdot)$ a $n = 2$, $F = \mathbb{C}$, získáme komplexní reprezentaci kvaternionů (zvolením $n = 4$, $F = \mathbb{R}$ bychom mohli najít vhodnou reálnou reprezentaci kvaternionů)

$$a + b\mathrm{i} + c\mathrm{j} + d\mathrm{k} \to \begin{pmatrix} a + b\mathrm{i} & c + d\mathrm{i} \\ -c + d\mathrm{i} & a - b\mathrm{i} \end{pmatrix}. \tag{3.9}$$

Opět se nejedná o jedinou možnou reprezentaci.

Nyní lze tuto reprezentaci rozšířit pro matice [40]. Kvaternionovou matici $\mathbf{M}$ lze rozepsat jako $\mathbf{M} = \mathbf{M}_1 + \mathbf{M}_2\mathrm{j}$, kde $\mathbf{M}_1, \mathbf{M}_2$ jsou komplexní matice. Reálná část $\mathbf{M}_1$ odpovídá reálné složce $\mathbf{M}$, imaginární část $\mathbf{M}_1$ odpovídá první imaginární složce $\mathbf{M}$, reálná část $\mathbf{M}_2$ odpovídá druhé imaginární složce $\mathbf{M}$ a imaginární část $\mathbf{M}_2$ odpovídá třetí imaginární složce $\mathbf{M}$. Poté komplexní reprezentace kvaternionové matice $\mathbf{M}$ vypadá následovně

$$\mathbf{M} \to \begin{pmatrix} \mathbf{M}_1 & \mathbf{M}_2 \\ -\overline{\mathbf{M}_2} & \overline{\mathbf{M}_1} \end{pmatrix}, \tag{3.10}$$

kde $\overline{\mathbf{M}_1}$ značí komplexně sdruženou matici k $\mathbf{M}_1$.

Jelikož homomorfismus zachovává operace, máme zajištěno, že násobení dvou kvaternionových matic je ekvivalentní s násobením jejich komplexních reprezentací.

Rozklad na vlastní vektory a čísla pro kvaternionové matice

Pro polynomy s kořeny v komplexním oboru $\mathbb{C}$ platí základní věta algebry. Tzn. že polynom s komplexními koeficienty stupně $n \geq 1$ má v komplexní rovině právě n kořenů. Pro polynomy s kořeny v $\mathbb{H}$ tato věta bohužel neplatí. Obecně platí, že polynom s kořeny v $\mathbb{H}$ má těchto kořenů nekonečně mnoho [40]. Příkladem je polynom $x^2 + 1$, který kromě kořenů $\mathrm{i}, -\mathrm{i}$ má například i kořeny $\frac{1}{2}\mathrm{i} + \frac{1}{2}\mathrm{j} + \frac{1}{\sqrt{2}}\mathrm{k}$, $\frac{1}{2}\mathrm{i} + \frac{1}{\sqrt{2}}\mathrm{j} + \frac{1}{2}\mathrm{k}$ atd.

Totéž platí i pro charakteristický polynom. Tedy matice v kvaternionovém oboru $\mathbb{H}$ má obecně nekonečně mnoho vlastních čísel. Navíc protože násobení kvaternionů není komutativní, výrazy $\mathbf{M}\mathbf{x} = \lambda\mathbf{x}$ a $\mathbf{M}\mathbf{x} = \mathbf{x}\lambda$ jsou rozdílné.

Definice 3.4. [14] *Kvaternion λ je pravé (levé) vlastní číslo kvaternionové matice $\mathbf{M}$, jestliže splňuje*

$$\mathbf{M}\mathbf{x} = \mathbf{x}\lambda \ (\mathbf{M}\mathbf{x} = \lambda\mathbf{x}). \tag{3.11}$$

Dále se pro potřeby této práce budeme zabývat pouze pravými vlastními čísly, tedy dále bude pojmem vlastní číslo chápan jako pravé vlastní číslo. Pro výpočet vlastních čísel zavedeme pojem *standardní vlastní čísla.*

Tvrzení 3.5. [40] *Každá $n \times n$ kvaternionová matice $\mathbf{M}$ má přesně n pravých vlastních čísel, která jsou komplexní čísla s nezápornou imaginární složkou. Tato vlastní čísla definujeme jako standardní vlastní čísla kvaternionové matice $\mathbf{M}$.*

Pro výpočet využijeme komplexní reprezentaci $\chi_{\mathbf{M}} \in \mathbb{C}^{2n\times 2n}$ kvaternionové matice $\mathbf{M} \in \mathbb{H}^{n\times n}$.

$$\chi_{\mathbf{M}} = \begin{pmatrix} \mathbf{M}_1 & \mathbf{M}_2 \\ -\overline{\mathbf{M}_2} & \overline{\mathbf{M}_1} \end{pmatrix}, \tag{3.12}$$

kde $\mathbf{M} = \mathbf{M}_1 + \mathbf{M}_2\mathrm{j}$ a $\mathbf{M}_1, \mathbf{M}_2 \in \mathbb{C}^{n\times n}$.

Tvrzení 3.6. [14] *Pro kvaternionovou matici $\mathbf{M} \in \mathbb{H}^{n\times n}$ jsou komplexní vlastní čísla $\mathbf{M}$ stejná jako vlastní čísla její komplexní reprezentace $\chi_{\mathbf{M}}$. Navíc se vlastní čísla $\chi_{\mathbf{M}}$ vyskytují v komplexně sdružených párech. Speciálně platí, má-li $\chi_{\mathbf{M}}$ jakékoliv reálné vlastní číslo, bude mít sudou násobnost. Tedy je možné pro kvaternionovou matici $\mathbf{M}$ získat n komplexních čísel s nezápornou imaginární složkou.*

Poté je vztah mezi vlastními vektory kvaternionové matice $\mathbf{M}$ a vlastními vektory její komplexní reprezentace $\chi_{\mathbf{M}}$ následovný: Pokud $(\mathbf{x})_{2n\times 1} = \begin{pmatrix} (\mathbf{x}_1)_{2n\times 1} \\ (\mathbf{x}_2)_{2n\times 1} \end{pmatrix}$ je vlastní vektor $\chi_{\mathbf{M}}$ příslušný vlastnímu číslu λ, tak $(\mathbf{x})_{n\times 1} = \mathbf{x}_1 - \overline{\mathbf{x}_2}\mathrm{j}$, $\mathbf{x}_1 \in \mathbb{H}^{n\times 1}$ je vlastní vektor kvaternionové matice $\mathbf{M}$ příslušný vlastnímu číslu λ.

SVD rozklad pro kvaternionové matice

Tvrzení 3.7. [14] *Pro každou kvaternionovou matici $\mathbf{M} \in \mathbb{H}^{m\times n}$ hodnosti r existují dvě unitární kvaternionové matice $\mathbf{U} \in \mathbb{H}^{m\times m}$ a $\mathbf{V} \in \mathbb{H}^{n\times n}$ takové, že platí*

$$\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^* = \mathbf{U}\begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}\mathbf{V}^*, \tag{3.13}$$

kde Σ_r je reálná diagonální matice obsahující r kladných hodnot na své diagonále a $\mathbf{V}^*$ značí hermitovskou transpozici matice $\mathbf{V}$.

Matice singulárních vektorů a singulárních čísel lze spočítat opět s pomocí komplexní reprezentace $\chi_{\mathbf{M}}$ kvaternionové matice $\mathbf{M}$. Postup je následovný:

Nejprve provedeme SVD rozklad komplexní matice $\chi_{\mathbf{M}} = \mathbf{U}_{\chi_{\mathbf{M}}}\Sigma_{\chi_{\mathbf{M}}}\mathbf{V}^*_{\chi_{\mathbf{M}}}$. Poté singulární čísla a vektory původní matice $\mathbf{M}$ získáme jako [14]:

$$\Sigma = \mathrm{row}_{\mathrm{odd}}(\mathrm{col}_{\mathrm{odd}}(\Sigma_{\chi_{\mathbf{M}}})) \tag{3.14}$$

$$\mathbf{U} = \mathrm{col}_{\mathrm{odd}}(\mathbf{U}_1) + \mathrm{col}_{\mathrm{odd}}(\overline{\mathbf{U}_2})\mathrm{j} \tag{3.15}$$

$$\mathbf{V} = \mathrm{col}_{\mathrm{odd}}(\mathbf{V}_1) + \mathrm{col}_{\mathrm{odd}}(\overline{\mathbf{V}_2})\mathrm{j}, \tag{3.16}$$

kde

$$\mathbf{U} = \begin{pmatrix} (\mathbf{U}_1)_{m\times 2m} \\ (\mathbf{U}_2)_{m\times 2m} \end{pmatrix}, \quad \mathbf{V} = \begin{pmatrix} (\mathbf{V}_1)_{n\times 2n} \\ (\mathbf{V}_2)_{n\times 2n} \end{pmatrix} \tag{3.17}$$

a $\mathrm{row}_{\mathrm{odd}}(\mathbf{A})$, $\mathrm{col}_{\mathrm{odd}}(\mathbf{A})$ představuje extrakci lichých řádků a sloupců matice $\mathbf{A}$.

Nyní můžeme přejít ke kvaternionovému rozšíření metod PCP a DMD. K tomu budeme jednotlivé pixely RGB obrazu reprezentovat pomocí ryzích kvaternionů tak, že

$$p = R \cdot \mathrm{i} + G \cdot \mathrm{j} + B \cdot \mathrm{k}, \tag{3.18}$$

kde p představuje barevný pixel a R, G, B jednotlivé barevné složky RGB obrazu. Tímto postupem zajistíme provázanost jednotlivých barevných složek.

3.2.2 Kvaternionové PCP

Tato metoda byla v článku [5] použita pro separaci audia. Jelikož algoritmus metody je stále stejný, pokusíme se ji použít i pro separaci videa.

Aby bylo možné rozšířit PCP metodu do kvaternionového oboru, je potřeba ukázat, jak rozšířit SVD rozklad a proximální operátory pro případ kvaternionů. Již byl ukázán SVD rozklad pro kvaternionové matice, tudíž je potřeba ještě provést rozšíření proximálních operátorů do prostoru $\mathbb{H}$.

Tvrzení 3.8. [5] *Proximální operátor* (2.13) *lze rozšířit do oboru kvaternionů* $\mathbb{H}$ *pomocí*

$$\mathrm{prox}_f(\mathbf{x}) = \arg\min_{\mathbf{y}} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + f(\mathbf{y}),\ \mathbf{y} \in \mathbb{H}. \tag{3.19}$$

Tedy proximální operátor pro ℓ_1 normu s parametrem λ vypadá následovně:

$$\mathrm{prox}_{\lambda\|\cdot\|_1}(\mathbf{x}) = \arg\min_{\mathbf{y}} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + \lambda\|\mathbf{y}\|_1,\ \mathbf{y} \in \mathbb{H}. \tag{3.20}$$

Vyjádříme-li řešení (3.20) po složkách získáme měkké prahování pro kvaterniony

$$\mathrm{soft}_\lambda(x_i) = y_i = \frac{x_i}{|x_i|}\max(|x_i| - \lambda, 0),\ \mathbf{y} \in \mathbb{H}. \tag{3.21}$$

Důkaz, že (3.21) je řešením proximálního operátoru (3.20) je k dispozici v článku [5].

Podobně jako v reálném případě nukleární normu lze nahradit ℓ_1 normou singulárních čísel, tedy proximální operátor nukleární normy (neboli singular value thresholding) pro kvaterniony vypadá následovně

$$\mathrm{svt}_\lambda(\mathbf{X}) = \mathrm{prox}_{\lambda\|\cdot\|_*}(\mathbf{X}) = \sum_{i=1}^{n} \mathbf{u}_i \mathrm{soft}_\lambda(\sigma_i)\mathbf{v}^*{}_i, \tag{3.22}$$

kde σ_i jsou singulární čísla a $\mathbf{u}_i, \mathbf{v}_i$ jsou singulární vektory matice $\mathbf{X} \in \mathbb{H}^{m \times n}$. Důkaz je opět k dispozici v článku [5].

Nyní můžeme definovat kvaternionové PCP jako

$$\min \ \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_1 \ \text{ za podmínky } \ \mathbf{L} + \mathbf{S} = \mathbf{M}, \tag{3.23}$$

kde $\mathbf{M} \in \mathbb{H}^{m \times n}$.

Problém (3.23) lze řešit stejným algoritmem jako klasické PCP (algoritmus 1), použijeme-li soft thresholding a singular value thresholding pro kvaterniony.

3.2.3 Kvaternionové DMD

Kvaternionové DMD [14] bude také rozšířením klasického DMD do oboru kvaternionů. Tzn. lze použít stejný algoritmus jako pro klasické DMD (algoritmus 3). Jedinou změnou v tomto algoritmu je, že původní matice dat a tudíž i matice $\mathbf{X}_1, \mathbf{X}_2$ z ní vzniklé jsou kvaternionové.

Jednotlivé kroky daného algoritmu jsme schopni provést, protože máme definované rozšíření SVD rozkladu a rozkladu na vlastní vektory a čísla pro kvaterniony. Jediné co neznáme, je definice exponenciální funkce pro kvaterniony. Tu nebude třeba definovat, protože celý algoritmus bude probíhat na komplexních reprezentacích daných kvaternionových matic. Tedy výsledné DMD mody budou komplexní.

Důvod, proč bude algoritmus proveden na komplexních reprezentacích daných matic, bude podrobněji rozebrán v další kapitole.

4 Implementace

V této kapitole bude rozebrána implementace jednotlivých metod. Všechny metody jsou implementovány v programu MATLAB R2021b. Volání všech níže popsaných metod je k dispozici v souboru `main.m`.

4.1 Načtení vstupních dat

Pro načtení vstupních dat jsou implementovány tři funkce.

První metoda `load_image(folder,format)` je určena k načítání série n po sobě jdoucích obrazů ve stupních šedi. Všechny obrazy se musí nacházet ve složce `folder` a mít příponu `format`. Přípona je očekávána ve tvaru `*.jpg`, `*.png` atd. Pokud jsou původní obrazy barevné, budou převedeny do stupňů šedi. Tato metoda vrací rozměry jednotlivých obrazů a matici vstupních dat $\mathbf{M}$, která byla popsána v podkapitole 2.1.

Druhou metodou je `load_color_image(folder,ycc,format)` a je určena pro načítání série n po sobě jdoucích barevných obrazů. Pokud je boolovský parametr `ycc=false`, obrazy se načtou v RGB. Pokud je `ycc=true`, obrazy jsou před načtením převedeny na $\mathrm{YC_BC_R}$ model. Výsledkem metody jsou opět rozměry jednotlivých obrazů a trojrozměrná matice vstupních dat, která byla opět popsána v podkapitole 2.1.

Poslední metodou je `load_video(name,rgb,ycc)`, která je určena pro načítání videí. Metoda z videa vytvoří sérii snímků, na které jsou zavolány výše zmíněné metody pro sérii n obrazů. Parametr `name` odpovídá jménu daného videa včetně přípony. Pokud jsou boolovské parametry `rgb=false` a `ycc=false`, video je načteno ve stupních šedi. Pokud je `rgb=true` a `ycc=false`, video je načteno v RGB prostoru a pokud `rgb=true` a `ycc=true`, video je načteno v $\mathrm{YC_BC_R}$.

4.2 Metody separace pro obrazy ve stupních šedi

Mediánový filtr

Metoda mediánového filtru je k nalezení v souboru `median_filtr.m`. Jak již bylo řečeno, implementace je velmi jednoduchá. K výpočtu mediánů jednotlivých řádků vstupní matice $\mathbf{M}$ je využita MATLAB funkce `median()`. Tím je získáno pozadí $\mathbf{L}$. Dynamická složka $\mathbf{S}$ je dopočítána jako rozdíl vstupních dat a pozadí $\mathbf{S} = \mathbf{M} - \mathbf{L}$.

PCP

Metoda PCP (algoritmus 1) je implementována v souboru `pcp.m`. Metoda obsahuje tři parametry: λ, μ a δ, které je potřeba nastavit podle vstupních dat.

SVD rozklad matice $\mathbf{L}$ je vypočítán pomocí MATLAB funkce `svd(L,'econ')` s vybranou možností `'econ'`, která představuje redukovaný singulární rozklad (poznámka 1.13). Redukovaný singulární rozklad je možné použít, protože nás zajímají pouze nenulová singulární čísla a singulární vektory, které tato singulární čísla násobí. Navíc singulární rozklad je výpočetně velmi náročná operace, je tedy lepší volit redukovanou verzi, která může snížit dobu výpočtu a nároky na úložiště.

Při výpočtu měkkého prahování singulárních čísel je možné vynechat výraz zachovávající znaménko, protože všechna singulární čísla jsou nezáporná reálná čísla. Tedy použité měkké prahování singulárních čísel vypadá následovně

$$\sigma_i^{k+1} = \max(\sigma_i^k - \frac{1}{\mu}, 0). \tag{4.1}$$

Jelikož matice $\mathbf{S}$ má i záporná čísla, měkké prahování jejích hodnot vypadá následovně

$$s_{i,j}^{k+1} = \frac{s_{i,j}}{|s_{i,j}|} \max(|s_{i,j}^k| - \frac{\lambda}{\mu}, 0). \tag{4.2}$$

Stop kritérium tohoto algoritmu je zvoleno podle [4] jako

$$\|\mathbf{M} - \mathbf{L}_{k+1} - \mathbf{S}_{k+1}\|_F > \delta \|\mathbf{M}\|_F, \tag{4.3}$$

kde parametr δ určuje s jakou přesností je separace provedena.

Nekonvexní RPCA

Byly vyzkoušeny dvě verze nekonvexního RPCA.

První se nachází v metodě `nonconvex_pcp_char(M,p)` a implementuje již dříve zmíněný algoritmus 2 navržený R. Chartrandem v [6]. Vstupní parametr p určuje zvolenou aproximaci nekonvexní ℓ_p normy, p tedy musí být z intervalu $\langle 0, 1\rangle$, kde $p = 1$ odpovídá jinému přístupu pro konvexní PCP metodu. Dále je potřeba opět podle vstupních dat nastavit parametry λ, μ a δ. Tato implementace stejně jako PCP využívá redukovaný singulární rozklad.

Protože singulární čísla jsou stále nezáporná, použité p-měkké prahování singulárních čísel vypadá následovně

$$\sigma_i^{k+1} = \max(\sigma_i^k - \mu(\sigma_i^k)^{p-1}, 0). \tag{4.4}$$

A p-měkké prahování pro prvky matice $\mathbf{S}$ je

$$s_{i,j}^{k+1} = \frac{s_{i,j}}{|s_{i,j}|} \max(|s_{i,j}^k| - \lambda\mu|s_{i,j}^k|^{p-1}, 0). \tag{4.5}$$

Dalším rozdílem tohoto nekonvexního algoritmu oproti PCP je, že se v každé iteraci zmenší parametr μ na $\frac{2}{3}$ své velikosti, dokud nedosáhne spodní hranice dané číslem $\mu_0 \cdot 10^{-7}$, kde μ_0 je počáteční zvolené μ. Tedy

$$\mu^{k+1} = \max\{\frac{2}{3}\mu^k, \mu_0 \cdot 10^{-7}\}. \tag{4.6}$$

Stop kritérium tohoto algoritmu je opět

$$\|\mathbf{M} - \mathbf{L}_{k+1} - \mathbf{S}_{k+1}\|_F > \delta\|\mathbf{M}\|_F. \tag{4.7}$$

V souboru `nonconvex_pcp.m` se nachází druhá verze implementace nekonvexního RPCA. Jedná se o algoritmus klasického PCP (algoritmus 1), pouze je místo měkkého prahování použito p-měkké prahování. Tato změna znamená, že místo Lagrangianu (2.46) je použit Lagrangian (2.17), ve kterém je norma ℓ_1 nahrazena proximální p-normou. Tedy p-měkké prahování singulárních čísel je

$$\sigma_i^{k+1} = \max(\sigma_i^k - \frac{1}{\mu}(\sigma_i^k)^{p-1}, 0) \tag{4.8}$$

a p-měkké prahování prvků matice $\mathbf{S}$ je

$$s_{i,j}^{k+1} = \frac{s_{i,j}}{|s_{i,j}|}\max(|s_{i,j}^k| - \frac{\lambda}{\mu}|s_{i,j}^k|^{p-1}, 0). \tag{4.9}$$

Vše další nastavení a implementace jsou stejné jako v PCP.

DMD

Implementace DMD (k nalezení v `dmd.m`) je částečně přebraná a dále upravená implementace použitá v knize [19]. Vstupní parametr ϵ určuje práh, kdy je ještě daný Fourierův mod brán jako pomalý. Další parametr, který je potřeba nastavit, je r určující zmenšení dimenze na pouze r dominantních modů.

Stejně jako v PCP je i zde proveden redukovaný singulární rozklad pomocí příkazu `svd(X1,'econ')`. Dále jsou vlastní čísla spočítána pomocí příkazu `eig()`. V této implementaci jsou kromě Fourierových a DMD modů pozadí spočítány i zbylé Fourierovy a DMD mody a z nich jsou vytvořeny obrazy popředí. Tyto obrazy ovšem nejsou použity jako dynamická složka. Slouží pouze k výpočtům pro Multiresolution DMD. Dynamická složka je dopočítána jako $\mathbf{S} = \mathbf{M} - |\mathbf{L}|$, čímž je zaručeno, že bude mít reálné hodnoty.

4.3 Metody separace pro barevné obrazy

Separace po jednotlivých složkách

Pokud chceme separaci pro barevné obrazy provést postupně po jednotlivých barevných složkách, můžeme k tomu využít již implementované metody pro obrazy

ve stupních šedi. Tedy vybranou metodu vložíme do cyklu, který se zopakuje třikrát – pro každou složku zvlášť. Výsledné složky uložíme do třetí souřadnice matic $\mathbf{L}, \mathbf{S}$. Tento formát je potřeba pro následné vykreslení a vyhodnocení.

Kvaternionové PCP

Kvaternionové PCP je implementováno v souboru `color_pcp.m`. Pokud bychom použili MATLAB toolbox – Quaternion[1], mohli bychom ponechat implementaci stejnou jako pro PCP pro obrazy ve stupních šedi pouze se změnou, že vstupní matice $\mathbf{M}$ bude kvaternionová. V této práci byl ovšem zvolen jiný přístup. Celý výpočet proběhne na komplexních reprezentacích kvaternionových matic (jak bylo ukázáno v podkapitole 3.2.1).

Tento postup byl zvolen převážně z důvodu výpočetní náročnosti. Vyzkoušíme-li celý postup na relativně malých datech, výpočet v komplexních reprezentacích trvá v rámci sekund a stačí na něj 10GB RAM paměti. Ale pro výpočet v kvaternionech trvá výpočet téměř hodinu – převážně z důvodu výpočtu SVD rozkladu kvaternionové matice. Navíc je k tomuto výpočtu potřeba až 60GB RAM paměti. A jelikož nejvíce výpočetně náročný úkon – SVD rozklad se opakuje v každé iteraci, je raději zvolen výpočet na komplexních reprezentacích.

Z matice vstupních dat vytvoříme kvaternionovou matici a následně její komplexní reprezentaci. Jelikož po SVD rozkladu pouze upravíme hodnoty singulárních čísel v $\mathbf{\Sigma}$ a získané matice $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}$ spolu opět vynásobíme (pro získání nové matice $\mathbf{L}$), nemusíme získané matice přepočítávat zpět do kvaternionů. Násobení matic komplexních reprezentací je ekvivalentní s násobením matic kvaternionů. Výsledek by tedy měl být přesný, i když bude celý výpočet ponechán v komplexních reprezentacích.

V průběhu celého algoritmu není žádný další výpočet, který by mohl změnit výsledky, ponecháme-li matice v komplexních reprezentacích. Všechny iterace až do zastavení probíhají tedy na komplexních reprezentacích a až po ukončení výpočtu jsou výsledné matice komplexních reprezentací převedeny zpět na kvaterniony (a ty na 3D matice $\mathbf{L}$,$\mathbf{S}$). Pro zpětné získání kvaternionů připomeňme, jak vypadá komplexní reprezentace

$$\mathbf{M} \to \begin{pmatrix} \mathbf{M}_1 & \mathbf{M}_2 \\ -\overline{\mathbf{M}_2} & \overline{\mathbf{M}_1} \end{pmatrix}, \tag{4.10}$$

kde $\mathbf{M} = \mathbf{M}_1 + \mathbf{M}_2 j$.

Tedy z komplexní reprezentace je možné získat původní kvaternionovou matici např. způsobem: Vezmeme horní polovinu řádků. První polovina jejich sloupců

[1] Dostupný z https://qtfm.sourceforge.io/

odpovídá matici $\mathbf{M}_1$. Tzn. reálná složka této čtvrtiny odpovídá reálné složce kvaternionové matice a imaginární složka odpovídá první imaginární složce kvaternionové matice. Druhá polovina jejich sloupců odpovídá matici $\mathbf{M}_2$. Tedy reálná složka této druhé čtvrtiny odpovídá druhé imaginární složce a imaginární složka této čtvrtiny odpovídá třetí imaginární složce kvaternionové matice.

Jelikož by všechny operace měly zachovat vztah mezi kvaternionovou maticí a její komplexní reprezentací, je možné obdobnou logikou kvaternionovou matici získat z dolní poloviny řádků (případně z jiné vhodné kombinace).

Kvaternionové DMD

Kvaternionové DMD bylo implementováno v `color_dmd.m`. V této implementaci probíhá výpočet za pomoci MATLAB toolboxu Quaternion v kvaternionových maticích. Jak již bylo řečeno, SVD rozklad na kvaternionech je výpočetně velmi náročná operace, proto je SVD rozklad proveden na komplexní reprezentaci a matice $\mathbf{U}, \Sigma, \mathbf{V}$ jsou zpětně přepočítány na kvaterniony podle předpisu z kapitoly 3.2.1.

Bohužel pro kvaterniony neexistuje implementace operace `A/B(A\B)`, proto místo funkce `A/B(A\B)` bude v celém algoritmu použito `A*inv(B)(inv(A)*B)`. Tento postup je sice náročnější a méně přesný, ale zdá se být jedinou možnou volbou.

Výpočet vlastních čísel a vektorů je opět proveden na komplexní reprezentaci, protože funkce `eig()` funguje jen pro hermitovské kvaternionové matice. Matice $\tilde{\mathbf{A}}$ bohužel obecně není hermitovská, výpočet tedy musí být proveden na komplexních reprezentacích. Z důvodu numerických chyb může nastat případ (při testování metody nastal často), že z rozkladu na vlastní vektory a čísla nezískáme přesnou polovinu vlastních čísel s nezápornou imaginární složkou – standardních vlastních čísel (viz tvrzení 3.5). Pokud některé číslo přebývá, odstraníme vlastní číslo s nejmenší imaginární složkou. Pokud některé vlastní číslo chybí, přidáme ke standardním vlastním číslům číslo s nejmenší velikostí záporné imaginární složky. Tyto postupy aplikujeme, dokud nezískáme přesně polovinu vlastních čísel, která považujeme za standardní.

V implementaci klasické DMD metody je pro výpočet vektoru počátečních koeficientů $\mathbf{b}$ použita funkce zpětného lomítka $\mathbf{b} = \Phi \backslash \mathbf{x}_1$, proto by se nabízelo použít inverzi matice Φ. Ta ale není regulární, je tedy potřeba použít pseudoinverzi. Proto vektor počátečních koeficientů dopočítáme jako $\mathbf{b} = \text{pinv}(\Phi) * \mathbf{x}_1$.

Zřejmě vlivem všech numerických chyb, které nastaly kvůli výše zmíněným důvodům, tato metoda nefunguje. Výsledné pozadí vychází zcela nesmyslně. Proto se pokusíme celou metodu provést také pouze na komplexních reprezentacích. Tato implementace se nachází v souboru `color_dmd_complex.dmd`.

Na začátku metody tedy vytvoříme komplexní reprezentace matic $\mathbf{X}_1, \mathbf{X}_2$ a veškeré další výpočty provádíme pouze na těchto komplexních reprezentacích. Tento postup je téměř funkční. Ale získaná výsledná matice pozadí $\mathbf{L}$ nemá ani zdaleka tvar komplexní reprezentace kvaternionové matice. Tzn. její jednotlivé čtvrtiny, které by měly být až na znaménko stejné, jsou silně rozdílné. Použijeme-li k rekonstrukci kvaternionové matice celou levou stranu získané komplexní reprezentace, získáme výsledek, který se jeví jako správný. Použijeme-li jakoukoliv část pravé strany, získáme v barevné složce nacházející se v této části nesmyslný výsledek.

Chybovost pravé strany je nejspíše způsobená výpočtem DMD modů

$$\mathbf{\Phi} = \mathbf{X}_2 \mathbf{V} \mathbf{\Sigma}^{-1} \mathbf{W}. \tag{4.11}$$

$\mathbf{W}$ je matice vlastních vektorů a tyto vlastní vektory, jak bylo ukázáno v podkapitole 3.2.1, jsou jiné než kvaternionové vektory. Je tedy možné, že chyba na pravé straně je způsobena těmito vektory. Jelikož není přesně jisté, co chybu pravé strany způsobuje, je to vhodný námět na další zkoumání.

4.4 Multiresolution DMD

Ve funkci `mrdmd(X,num,i,ii,omega,phi,eps1,eps2)` je implementována mrDMD metoda. Tato funkce se volá rekurzivně pro jednotlivé úseky vstupního videa `X`. Proto je potřeba držet si v paměti, na které dekompoziční úrovni se momentálně nacházíme. K tomu slouží proměnná `i`. Pokud je `i=1`, nacházíme se na první úrovni, tzn. separujeme pozadí. Dále je potřeba držet v paměti, ve kterém časovém segmentu se právě nacházíme. K tomu slouží proměnná `ii`, jedná se o vektor délky `num` – požadovaný celkový počet dekompozičních úrovní. Každý prvek `ii` nabývá hodnot 0 až $2^{\text{num}-1}$ – počet časových segmentů, na které je rozdělena poslední dekompoziční úroveň. Na dekompoziční úrovni `i` je poté jeden časový krok získán jako $2^{\text{num}-1}/2^{\text{i}-1}$. Jinými slovy toto číslo určuje, kolik časových dílků na poslední dekompoziční úrovni tvoří jeden časový dílek na úrovni `i`. Přičtením správného časového kroku k hodnotě na `i`-tém místě vektoru `ii` se posouváme po jednotlivých časových segmentech.

Tato funkce funguje tak, že pro matici vstupních dat `X` zavolá funkci `dmd(X,eps)`, která spočítá pomalé Fourierovy mody, pomalé DMD mody a matici, která je získána výpočtem obrazu reprezentujícího zbylé rychlejší mody. Získané mody poté zapíše do 3D matice `omega` a 4D matice `phi`. Vektor získaných Fourierových modů je zapisován do proměnné `omega` jako sloupec, jehož druhá souřadnice odpovídá dekompoziční úrovni `i` a jeho třetí souřadnice odpovídá danému časovému segmentu (`i`-tý prvek vektoru `ii`). Matice získaných DMD modů je zapisována do proměnné `phi` jako matice s třetí souřadnicí odpovídající úrovni `i` a čtvrtou souřadnicí odpovídající časovému segmentu. Matice odpovídající rychlejším modům je poté rozdělena

na poloviny a na každou tuto polovinu je rekurzivně zavolána opět funkce `mrdmd()`. Takto je pokračováno, dokud se neprojdou všechny časové segmenty všech dekompozičních úrovní.

Metoda má dva parametry `eps1,eps2`, které slouží jako prahy pro funkci `dmd()` rozhodující, kdy jsou mody ještě pomalé a kdy už nikoliv. V této implementaci jsou tyto parametry voleny tak, že pro první dekompoziční úroveň je použit práh `eps1` a pro zbylé dekompoziční úrovně je použit práh `eps2`. Prahy jsou takto voleny, protože velikost Fourierova modu odpovídajícího pozadí je zpravidla o několik řádů menší než velikost ostatních. Proto je pro správnou extrakci pozadí potřeba menšího prahu než pro extrakci dalších pomalejších složek.

Volání metody `mrdmd()` je zabaleno ve funkci `mrdmd_result(X,num,eps1,eps2)`, která zároveň pro každou dekompoziční úroveň spočítá výslednou matici reprezentující pomalé mody dané úrovně. Tuto matici poté uloží do třetí souřadnice (odpovídající úrovni) výsledné matice `X_res`. Matice reprezentující zbylé rychlé mody je dopočítána jako rozdíl matice vstupních dat a součtu matic všech extrahovaných pomalejších složek. Tedy výsledkem funkce `mrdmd_result()` je 3D matice, jejíž jednotlivé 2D matice reprezentují složky extrahované na každé úrovni až po matici obsahující zbylé rychlé složky.

4.5 Vykreslení

Aby bylo možné zhodnotit přesnost výsledků, vykreslíme výsledky jako video, které je synchronním sloučením tří videí – první vykresluje původní vstupní data, pod ním se nachází video vykreslující získanou nízkohodnostní složku (pozadí) a poslední je video vykreslující získanou řídkou (dynamickou) složku.

Obrazy ve stupních šedi

Metoda provádějící vykreslení neupravených dat získaných separací se nachází v souboru `vykresleni.m`.

Bohužel separace vstupních dat způsobuje, že získaná řídká složka je tmavší než by měla být. Může dokonce dosáhnout záporných hodnot. To je problém, protože pro vykreslení je potřeba jednotlivé složky převést do formátu `uint8`, který všechny záporné hodnoty zaokrouhlí na nulu – budou tedy černé. Z tohoto důvodu je pro potřeby vykreslení dynamická složka upravena.

Podrobnější vysvětlení, proč dochází ke ztmavení dynamické složky, je rozebráno v bakalářské práci [10]. V práci [10] jsou také navrženy způsoby, jak dynamickou složku co nejvěrohodněji vizualizovat. Metody s nejlepšími výsledky jsou škálování jednotlivých obrazů a filtrace jednotlivých obrazů. Škálování je implementováno v

souboru `skalovane_vykresleni.m`. Touto metodou je nejmenší hodnota v každém obrazu přeškálována na číslo 0 a největší hodnota na číslo 255.

Filtrace obrazů je implementována v souboru `filtr_vykresleni.m`. Při tomto postupu jsou nejprve všechny hodnoty menší nebo rovny 4 zaokrouhleny na 0. Poté je každý obraz filtrován Wienerovým filtrem [46] a hodnoty menší nebo rovny 4 jsou opět zaokrouhleny na 0. Poté je na celý obraz aplikováno morfologické uzavření [42].

Detailní popis výše zmíněných postupů spolu s výhodami a nevýhodami jednotlivých metod vykreslení je k dispozici v bakalářské práci [10].

Barevné obrazy

Neupravený výsledek separace barevných dat je možné vykreslit pomocí funkce nacházející se v souboru `color_vykresleni.m`.

I pro barevné obrazy je z důvodu lepšího vyhodnocení výsledků provedena úprava dynamické složky. Všechny úpravy jsou stejné jako pro obrazy ve stupních šedi, pouze jsou aplikovány na každou barevnou složku zvlášť.

Škálování nejmenšího prvku na 0 a největšího na číslo 255 je k dispozici v souboru `color_skalovane_vykresleni.m`. Filtrace pomocí Wienerova filtru je k dispozici v souboru `color_filtr_vykresleni.m`.

Multiresolution DMD

Vykreslení výsledných složek získaných pomocí mrDMD je provedeno stejně jako pro obrazy ve stupních šedi s rozdílem, že videa představující výsledky pro jednotlivé složky jsou poskládány vedle sebe. Tedy první video se skládá z videa vstupních dat, pozadí (první získané pomalejší složky) a druhé získané pomalejší složky. Druhé video tvoří video vstupních dat, pozadí a třetí získané pomalejší složky. A poslední video tvoří video vstupních dat, pozadí a zbylé rychlé složky.

Opět je v souboru `skalovane_mrdmd_vykresleni.m` implementována škálovaná verze vykreslení a v souboru `filtr_mrdmd_vykresleni.m` je implementováno vykreslení využívající Wienerův filtr a morfologické uzavření.

5 Porovnání výsledků

Všechny výsledky zmíněné v této kapitole se nacházejí ve složce `vysledky`.

5.1 Implementace vyhodnocení výsledků

Aby bylo možné změřit přesnost separace, bude potřeba vyrobit simulovaná data. Ta je možné vyrobit např. pomocí souboru `data_maker.m`, který vezme jeden obraz jako pozadí a další dva obrazy jako dynamickou složku. Tyto dva obrazy dynamické složky jsou poté náhodně posouvány po obrazu pozadí.

Provedeme-li separaci pro takto získaná simulovaná data, můžeme získaný obraz pozadí porovnat s původním obrazem pozadí, ze kterého byla simulovaná data vytvořena. Porovnání získaného a původního obrazu pozadí ve stupních šedi je implementováno v souboru `vyhodnoceni.m`. Porovnání probíhá pomocí Frobeniovy normy rozdílu získaného a původního pozadí a pomocí tzv. *SSIM indexu* [45].

Jedná se o index vyjadřující podobnost obrazů. Může nabývat hodnot z intervalu $\langle -1, 1 \rangle$, kde hodnota 1 značí dva zcela identické obrazy, hodnota -1 značí zcela rozdílné obrazy a hodnota 0 značí nulovou strukturální podobnost. Pro porovnávání bude použita i tzv. *SSIM mapa*, kterou nabízí program MATLAB. Tato funkce vykreslí lokální hodnoty SSIM pro celý obraz. Pro lepší viditelnost výsledků budou hodnoty SSIM mapy přeškálovány tak, aby černá barva odpovídala hodnotě 0,85 (nejmenší SSIM hodnota, která byla naměřena) a bílá hodnotě 1.

Vyhodnocení pro barevné obrazy používá taktéž Frobeniovu normu a SSIM index. Tyto metriky jsou aplikovány na každou barevnou složku zvlášť. Je tedy získána přesnost každé barevné složky. Vyhodnocení pro barevné obrazy je implementováno v souboru `color_vyhodnoceni.m`.

5.2 Volba parametrů

Volba parametrů bude určena převážně experimentálně. K tomu budou užita simulovaná data vzniklá z obrazů 5.1, u kterých je možné porovnat získané pozadí s původním.

Získaný obraz pozadí bude porovnán s původním pozadím pomocí Frobeniovy normy a SSIM indexu. Dále bude *max rozdíl* vyjadřovat největší hodnotu Frobeniovy normy rozdílu mezi získanými obrazy pozadí a původním pozadím, *prům. rozdíl* průměrnou hodnotu Frobeniovy normy rozdílu mezi získanými obrazy pozadí a původním pozadím, *min SSIM* nejmenší naměřený SSIM index mezi získanými obrazy pozadí a původním pozadím a *prům. SSIM* průměrnou hodnotu SSIM indexu naměřenou mezi získanými obrazy pozadí a původním pozadím.



(a) Dynamická složka (b) Dynamická složka (c) Pozadí

Obr. 5.1: Simulovaná vstupní data. Dynamická složka je pro lepší viditelnost zobrazena větší, než tomu je ve skutečnosti. Objekt (b) se po pozadí pohybuje šestkrát rychleji než objekt (a).

DMD

V DMD metodě je možné volit dva parametry – práh ϵ a parametr redukující dimenzi r.

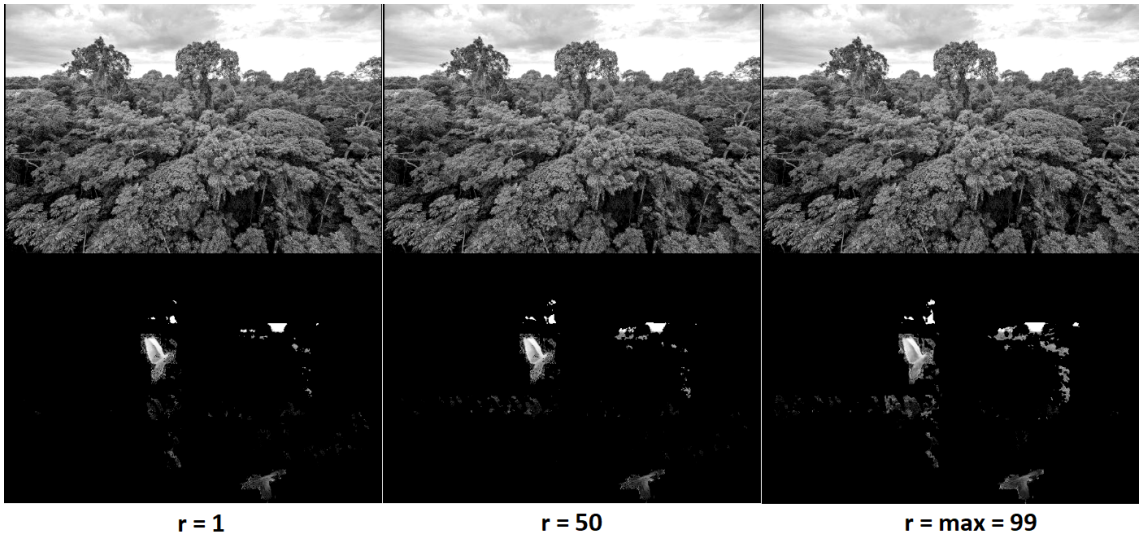
Experimentálními výsledky na simulovaných i reálných datech se ukazuje, že velikost Fourierova modu, který náleží pozadí, je na rozdíl od zbylých modů velmi malá. Řádově se pohybuje od 10^{-6} do 10^{-3}. Velikost zbylých modů je většinou řádově 10^{-1} výjimečně 10^{-2}. Proto jako parametr ϵ bude voleno číslo 0,01.

Parametr r by měl být volen tak, aby zachytil všechny dominantní mody. U simulovaných dat víme, že správná hodnost matice pozadí $\mathbf{L}$ by měla být jedna, protože série obrazů vznikla z jediného obrazu pozadí. Tedy v tomto případě by měla být ideální volba $r = 1$. Tento fakt zobrazuje tabulka 5.1, kde byly porovnány výsledky s použitím $r = 1$, což odpovídá nejmenší možné volbě r, dále s $r = 50$ a $r = 99$, což odpovídá maximální možné volbě r (původních obrazů je 100). Tyto výsledky jsou také vykresleny na obrázku 5.2 a k nalezení jsou ve složce `simulovana_data\grayscale\dmd`. Z obrázku 5.2 je zřetelné, že čím je r větší, tím větší chyba je zanášena do dynamické složky. Konkrétně se do dynamické složky promítají jednotlivá místa, kde se vyskytovala dynamická složka v průběhu všech snímků.

Jelikož u reálných videí nevíme, kolik dominantních modů je potřeba, abychom vystihli chování daného systému, budeme pro reálná videa vždy volit maximální možné r. Kdybychom zvolili r menší, může nastat, že vynecháme důležitý mod, což způsobí značně větší nepřesnost, než která vznikne, zvolíme-li maximální možné r. Tedy $r = n - 1$, kde n značí počet snímků daného videa.

Tab. 5.1: Porovnání výsledků DMD metody pro různé volby parametru r. Původních obrazů je 100, $r = 99$ je tedy maximální možná volba r a $r = 1$ je minimální možná volba. Výsledné snímky pozadí jsou porovnány s originálním pozadím a podobnost je zhodnocena pomocí Frobeniovy normy, SSIM indexu a časové náročnosti.

Výsledky DMD metody pro různé volby parametru r.					
	čas$[s]$	max rozdíl	prům. rozdíl	min SSIM	prům. SSIM
$r = 1$	1,3	1961,0	1957,8	0,9778	0,9778
$r = 50$	2,8	1971,5	1968,7	0,9771	0,9771
$r = 99$	3,6	2010,8	2008,3	0,9767	0,9767



Obr. 5.2: Výsledek DMD metod pro data ve stupních šedi pro různé hodnoty parametru r. V horní polovině je vykresleno odseparované pozadí, v dolní polovině je vykreslena získaná dynamická složka. Obrazy dynamické složky jsou vykresleny pomocí filtrovaného vykreslení. Obrazů je 100, tzn. největší možná volba parametru r je 99.

PCP

V PCP metodě je potřeba správně nastavit dva parametry λ, μ a přesnost δ. Parametr μ budeme volit stejně jako autoři článku [4] a to $\mu = m \cdot n/(4\|\mathbf{M}\|_1)$, kde m, n jsou rozměry matice vstupních dat $\mathbf{M}$. Parametr μ volíme takto pevně, protože špatná volba může způsobit, že se smyčka PCP algoritmu zacyklí. Jelikož pro větší data je výpočet každé iterace časově náročný, raději zvolíme μ pevně a zaměříme se na parametr λ.

Jak již bylo dříve řečeno, pro λ existuje univerzální volba $\lambda = \frac{1}{\sqrt{(\max(m,n))}}$, kde

m, n jsou rozměry matice vstupních dat. Pro simulovaná data je tato univerzální $\lambda = 0,002$. Pokud jsme zvolili $\lambda \leq 0,001$ nebo $\lambda \geq 0,008$, bylo pomocí Frobeniovy normy a SSIM indexu zjištěno, že čím menší v prvním případě a čím větší v druhém případě λ bylo, tím méně přesná byla separace. Pro $0,001 < \lambda < 0,008$ je podle metrik výsledek separace stejný. Jediný rozdíl je v počtu potřebných iterací. Ten je nejmenší právě pro volbu $\lambda = 0,002$ a její blízké okolí. Jelikož i pro reálná data má tento univerzální parametr velmi dobré výsledky, bude použitý k výpočtu výsledků určených k porovnání s výsledky dalších metod.

Přesnost bude v této práci dále volena jako $\delta = 10^{-5}$. Pokud bylo zvoleno $\delta = 10^{-6}$, nastaly situace, kdy se algoritmus zacyklil. I v případech, kdy se nezacyklil, trval zpravidla o několik desítek iterací více než při volbě $\delta = 10^{-5}$. Přitom získaný výsledek byl shodný jako při užití $\delta = 10^{-5}$. Pokud bylo zvoleno např. $\delta = 10^{-4}$, byla touto volbou ovlivněna přesnost separace. Proto i nadále bude používáno $\delta = 10^{-5}$.

Nekonvexní RPCA

Při užití nekonvexního RPCA implementovaného v souboru `nonconvex.m` jsme byli schopni pro simulovaná data a pro všechny volby $p \in \langle 0, 1)$ najít kombinaci parametrů λ, μ, pro které tento algoritmus konverguje a získané výsledky dávají smysl. Bohužel pro reálná data jsme toho byli schopni pouze pro $p \geq 0,8$. Pro menší p byl tento úkon výpočetně velmi náročný. Správný výsledek pro $p = 0,8$ byl získán až po 635 iteracích. Bylo tedy potřeba nechat daný algoritmus běžet alespoň 1000 iterací. Bohužel u většiny kombinací daných parametrů se algoritmus zacyklil – dosáhl 1000+ iterací. Vzhledem k časové náročnosti jedné iterace (způsobené SVD rozkladem) bylo rozhodnuto tento algoritmus dále nepoužívat. Proto se dále zaměříme pouze na původní algoritmus 2 dle Chartranda.

V tomto algoritmu je opět potřeba nastavit dva parametry λ, μ a přesnost δ. Přesnost bude ze stejných důvodů jako v PCP algoritmu volena jako $\delta = 10^{-5}$.

Počáteční hodnota parametru μ je v článku [6] volena jako $4/5$ velikosti největšího singulárního čísla matice vstupních dat. I v této práci se pokusíme μ takto ponechat. Případné další volby μ budou vždy násobky velikosti největšího singulárního čísla matice vstupních dat. Přesnou volbu λ, μ nalezneme pro daná vstupní data experimentálně.

Kvaternionové PCP

Jelikož kvaternionové PCP probíhá na komplexních reprezentacích bez jakékoliv změny původního PCP algoritmu, budou i parametry ponechány stejné jako v PCP algoritmu.

Kvaternionové DMD

Kvaternionové DMD probíhá stejně jako klasické DMD, proto práh ϵ ponecháme stejný jako v DMD. Ovšem pro volbu parametru r bylo experimentálně zjištěno, že i v případě simulovaných dat volba $r = 1$ nevrací smysluplný výsledek. Proto pro kvaternionové DMD budeme vždy používat největší možnou volbu $r = n - 1$, kde n je počet sloupců matice vstupních dat $\mathbf{M}$.

Multiresolution DMD

V mrDMD metodě je opět potřeba volit parametr r redukující dimenzi, počet dekompozičních úrovní L a práh ϵ. Parametr r bude volen stejně jako v klasickém DMD (ze stejných důvodů) jako maximální možná volba $r = n - 1$, kde n je počet sloupců matice vstupních dat.

Počet dekompozičních úrovní L je třeba volit podle vstupních dat. Budeme-li mít video, kde budou rychle jezdit auta, pomaleji kola a nejpomaleji se budou pohybovat chodci, vyžadujeme tři úrovně dekompozice (1. pozadí, 2. chodci a 3. kola). Přičemž jedoucí auta by měla zůstat jako zbylé rychlé mody. Podobnou logikou volíme počet dekompozičních úrovní i pro jiná vstupní data.

Práh ϵ budeme volit menší pro první dekompoziční úroveň a větší pro zbylé úrovně. Důvodem je, jak již bylo řečeno, že velikost Fourierových modů odpovídajících pozadí (1. dekompozice) je mnohem menší než velikost ostatních modů. Abychom při volbě $\epsilon = 0,001$ byli schopni odseparovat složky v dalších úrovních, musely by dané složky vypadat v daném časovém segmentu opravdu jako pozadí – musely by být zcela statické. Toho v reálném případě většinou nejsme schopni dosáhnout (museli bychom vzít např. pouze 3 snímky), proto pro další úrovně budeme volit jiné ϵ, které je potřeba pro daná vstupní data najít experimentálně. Podle potřeby je možné volit ϵ jako funkci závislou na dekompoziční úrovni i pro další úrovně.

Úspěšnost mrDMD ilustrujeme na videu ulice `street.mp4`[1]. Na toto video aplikujeme klasickou DMD metodu i mrDMD. Ukázka z výsledku separace je vykreslena na obrázku 5.3. Z obrázku je viditelné, že DMD metoda do dynamické složky kromě projíždějícího auta přidá i část ulice a budov. Na tomto videu se kromě aut a člověka pohybuje i světlo vrhané danými auty. Lidské oko toto světlo téměř nezaznamená. DMD metoda ho přidá do dynamické složky, ale pomocí mrDMD metody jsme schopni toto světlo odseparovat. Konkrétně použijeme-li práh k získání pozadí $\epsilon = 0,002$ a práh k získání dalších složek $\epsilon = 0,4$, tak pohybující se světlo je získáno z druhé dekompoziční úrovně mrDMD algoritmu. Popsané výsledky se nacházejí ve složce `street`.

[1] Dostupné z `https://www.pexels.com/video/man-crossing-the-streets-of-london-5266869/`

Obr. 5.3: Rozdíl mezi výsledky mrDMD a DMD metod. V horní třetině je vykreslen původní obraz, v druhé třetině je vykresleno odseparované pozadí a v dolní třetině je vykreslen obraz získaný z druhé dekompoziční úrovně mrDMD metody a obraz dynamické složky pro DMD metodu. Dynamická složka je vykreslena pomocí škálovaného vykreslení. Je patrné, že DMD metoda odseparovala nejen projíždějící auto, ale i část budov a silnice. Budovy a silnice byly odseparovány, protože se po nich pohybuje světlo jedoucího automobilu. Pohybující se světlo není v původním videu na první pohled viditelné, ovšem pomocí mrDMD (konkrétně z druhé dekompoziční úrovně) jsme toto světlo schopni odseparovat.

5.3 Výsledky na simulovaných datech

Nejprve budou metody porovnány na simulovaných datech obsahujících 100 různých snímků o velikosti 600×399 px. Výpočty probíhají na notebooku s procesorem AMD Ryzen 5 5600H, 3,30 GHz a 16GB RAM. Snímky simulovaných dat jsou dostupné ve složce `imgs`. Originální obrazy 5.1, ze kterých byla simulovaná data vytvořena, jsou uloženy ve složce `original_data` pod názvy `parot.png`, `parot2.png` a `forest.jpg`.

Nejprve se soustředíme na metody pro data ve stupních šedi. Všechny dále zmíněné výsledky se nacházejí ve složce `simulovana_data\grayscale`.

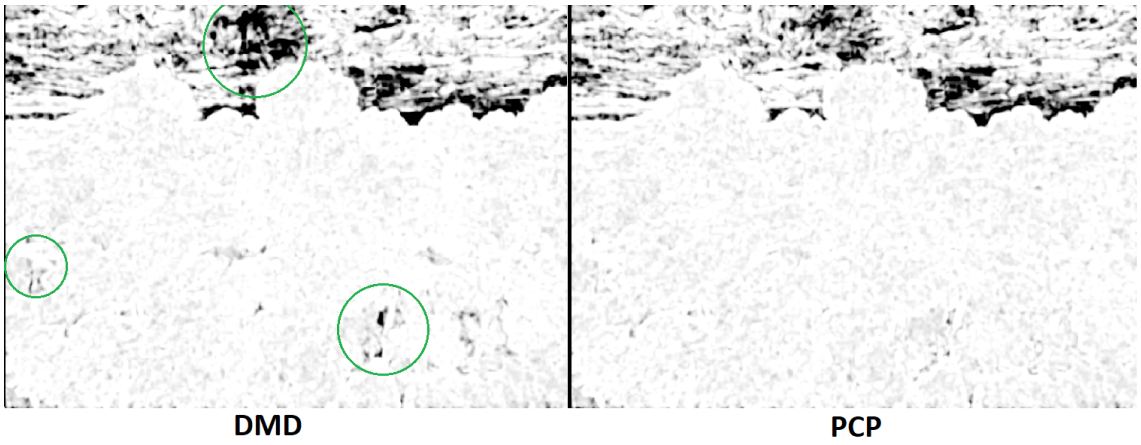
Tab. 5.2: Porovnání výsledků konvexních metod na simulovaných datech. Výsledky jsou porovnány výpočtem časové náročnosti, hodnosti získané matice pozadí $\mathbf{L}$ a pomocí ℓ_1 normy matice dynamické složky $\mathbf{S}$. Metoda PCP zkonvergovala po 25 iteracích a konvexní RPCA po 30 iteracích.

Výsledky konvexních metod na simulovaných datech.			
	čas$[s]$	rank($\mathbf{L}$)	$\|\mathbf{S}\|_1 \cdot 10^7$
Mediánový filtr	0,3	1	1,9400
DMD	1,3	1	3,5167
PCP	25,4	1	1,9400
RPCA s $p = 1$	36,8	2	1,9403

Simulovaná data nejprve separujeme pomocí konvexních metod – mediánový filtr, DMD, PCP a RPCA dle Chartranda s konvexní volbou $p = 1$. Výsledná hodnost matice $\mathbf{L}$ a ℓ_1 norma matice $\mathbf{S}$ jsou spolu s časovou náročností pro jednotlivé metody rozepsány v tabulce 5.2. Z této tabulky je patrné, že kromě RPCA všechny metody získaly rank($\mathbf{L}$) roven jedné, což je ideální výsledek, protože pozadí obrazů bylo vytvořeno z jediného snímku. Pouze RPCA metoda měla rank($\mathbf{L}$) = 2. Podíváme-li se ale na ℓ_1 normu získané matice $\mathbf{S}$, zdá se, že i přes větší hodnost $\mathbf{L}$ by mohla mít konvexní RPCA metoda lepší výsledek než metoda DMD. Tento fakt ověříme pomocí Frobeniovy normy rozdílu získaných obrazů pozadí s původním obrazem pozadí a pomocí SSIM indexu mezi získanými obrazy pozadí a původním obrazem. Konkrétní výsledky jsou k dispozici v tabulce 5.3.

Z tabulky 5.3 je patrné, že metody mediánový filtr, PCP i RPCA získaly stejně přesné obrazy pozadí. Dokonce musely být všechny obrazy získané RPCA metodou stejně přesné, i když získané pozadí musely tvořit alespoň dva různé obrazy. Pozadí získané pomocí DMD metody ale bylo méně přesné. Konkrétní problém je ukázán na obrázku 5.5. Do dynamické složky byly přidány poloprůhledné obrazy dynamické složky. Vzhledem k tomu, že dynamická složka je v DMD metodě spočítána jako $\mathbf{S} = \mathbf{M} - |\mathbf{L}|$, musí být stejný problém i součástí pozadí.

Tvrzení, že do obrazu pozadí byla DMD metodou přenesena i chyba na místech, kde se nacházela dynamická složka, je možné ověřit z obrázku 5.4. Na tomto obrázku jsou zeleně zvýrazněna místa, kde je větší chybovost DMD metody oproti PCP. Při porovnání s originálním videem je zřejmé, že se jedná o místa, kde se vyskytovala dynamická složka.



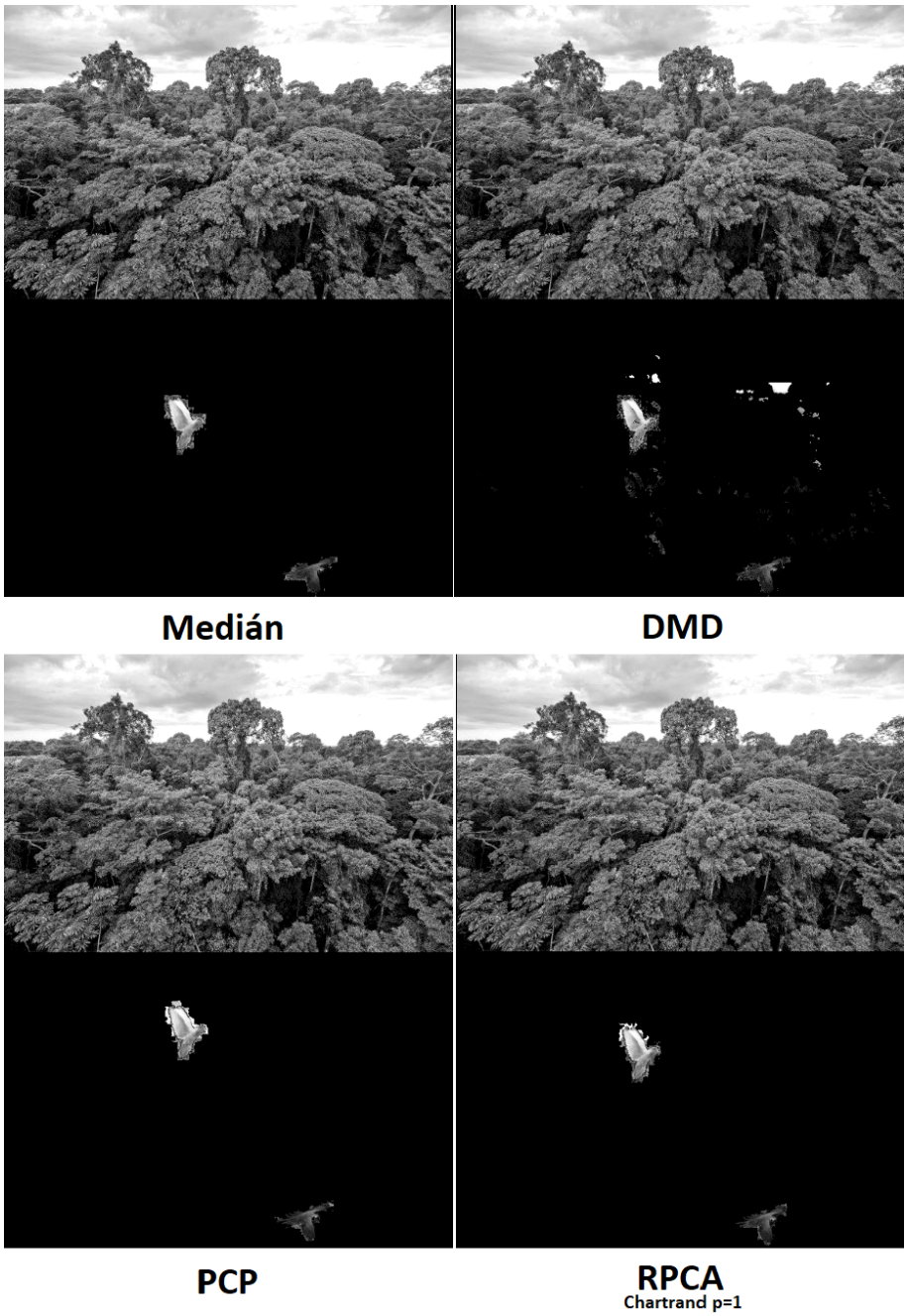
Obr. 5.4: Porovnání SSIM mapy DMD a PCP metody. Zelenými kroužky jsou zvýrazněna hlavní místa, kde je obraz pozadí získaný DMD metodou méně přesný než obraz získaný pomocí PCP. Jedná se o místa, kde byla dynamická složka.

Tab. 5.3: Porovnání výsledků konvexních metod na simulovaných datech. Výsledné snímky pozadí jsou porovnány s originálním pozadím a podobnost je zhodnocena pomocí Frobeniovy normy a SSIM indexu.

Výsledky konvexních metod na simulovaných datech.				
	max rozdíl	prům. rozdíl	min SSIM	prům. SSIM
Mediánový filtr	1929,9	1929,9	0.9799	0,9799
DMD	1961,0	1957,8	0,9778	0,9778
PCP	1929,9	1929,9	0,9799	0,9799
RPCA s $p = 1$	1929,9	1929,9	0,9799	0,9799

Dále vyzkoušíme, jestli nekonvexní varianty RPCA nedosáhnou přesnějších nebo rychlejších výsledků. Pro každou volbu p bylo potřeba experimentálně najít parametry λ a μ. Ty byly vybrány tak, aby matice $\mathbf{L}$ měla co nejmenší hodnost a $\|\mathbf{S}\|_1$ byla co nejmenší. Výsledky jsou rozepsány v tabulce 5.4. Z této tabulky je patrné, že výpočet byl náročnější než pro konvexní verzi tohoto algoritmu. Spočítáme-li přesnost separace pomocí Frobeniovy normy a SSIM indexu, získáme stejné výsledky jako pro mediánový filtr, PCP a konvexní RPCA. Můžeme tedy říct, že nejlepší metodou pro simulovaná data ve stupních šedi se jeví být mediánový filtr – je nejrychlejší a dosahuje stejně dobrých výsledků jako složitější metody.

Nyní se soustředíme na barevnou verzi simulovaných dat. Dále zmíněné výsledky se nacházejí ve složce `simulovana_data\color`. Postupně tedy vyzkoušíme kvaternionové DMD (QDMD), kvaternionové PCP (QPCP), výpočet DMD a PCP zvlášť pro jednotlivé barevné složky a nakonec převedeme data do $\mathrm{YC_BC_R}$ a opět spočí-



Obr. 5.5: Porovnání výsledků jednotlivých konvexních metod pro data ve stupních šedi získaných na simulovaných datech (konkrétně se jedná o snímek `img033.jpg`). V horní polovině obrazů výsledků pro jednotlivé metody je vykresleno odseparované pozadí, v dolní polovině je vykreslena získaná dynamická složka. Obrazy dynamické složky jsou vykresleny pomocí filtrovaného vykreslení. Metoda značená jako RPCA odpovídá konvexní variantě algoritmu 2, tzn. s volbou $p = 1$.

táme DMD a PCP zvlášť pro jednotlivé barevné složky. Výsledky opět porovnáváme podle časové náročnosti a pomocí Frobeniovy normy a SSIM indexu pro jednotlivé

Tab. 5.4: Výsledky nekonvexního RPCA pro různé hodnoty p na simulovaných datech. Výsledky jsou porovnány výpočtem hodnosti získané matice pozadí $\mathbf{L}$ a pomocí ℓ_1 normy matice dynamické složky $\mathbf{S}$. Parametry λ a μ_0 byly nalezeny experimentálně. Parametr μ_0 (počáteční hodnota parametru μ) je udáván v násobcích největšího singulárního čísla $\sigma_{\max}$ matice vstupních dat $\mathbf{M}$.

Výsledky nekonvexního RPCA na simulovaných datech.						
	λ	$\mu_0 \cdot \sigma_{\max}$	čas$[s]$	poč. iterací	rank($\mathbf{L}$)	$\|\mathbf{S}\|_1 \cdot 10^7$
0,9	0,001	4/5	50,2	30	2	1,9402
0,8	0,0019	4/5	51,3	29	2	1,9403
0,7	0,0025	4/5	51,1	29	2	1,9403
0,6	0,0005	10	53,2	30	1	1,9404
0,5	0,0007	10	57,3	32	2	1,9402
0,4	0,00015	50	61,3	34	2	1,9403
0,3	0,00015	50	60,0	33	2	1,9403
0,2	0,00015	100	60,1	33	2	1,9403
0,1	0,00015	150	61,5	34	2	1,9402
0	0,00015	1000	43,5	36	1	1,9403

barevné složky. Výsledek je v tabulce 5.5.

Z tabulky 5.5 je zřejmé, že jednotlivé druhy PCP metod mají lepší výsledky než odpovídající DMD verze. Tento fakt je viditelný i na obrázku 5.6. Nejlepších výsledků je dosaženo při užití QPCP a PCP provedeného po jednotlivých složkách RGB dat. Jelikož výpočet QPCP trvá téměř osmkrát tak dlouho jako výpočet PCP pro jednotlivé složky, jeví se PCP po jednotlivých barevných RGB složkách jako nejvhodnější metoda pro daná barevná simulovaná data.

Výsledky získané PCP a DMD metodami provedenými po jednotlivých $\mathrm{YC_BC_R}$ složkách nebudeme nadále vykreslovat. Důvod tohoto rozhodnutí je vyobrazen na obrázku 5.7. Dynamická složka je separována tak, aby byla řídká – obsahovala co nejvíce nul. Pokud pixel v $\mathrm{YC_BC_R}$ má hodnotu $[0, 0, 0]$ při převedení do RGB má hodnotu $[0, 128, 0]$, je tedy zelený. Proto pro srovnání tohoto přístupu s jinými metodami budeme používat pouze hodnost matice $\mathbf{L}$.

Tab. 5.5: Porovnání výsledků metod pro barevná data na simulovaných datech. Výsledné snímky pozadí jsou pro jednotlivé barevné složky porovnány s originálním pozadím a podobnost je zhodnocena pomocí Frobeniovy normy, SSIM indexu a časové náročnosti. Výsledky metod probíhajících v YC_BC_R byly převedeny zpět do RGB.

Výsledky metod pro barevná data na simulovaných datech.						
	QDMD	DMD (RGB)	DMD (YC_BC_R)	QPCP	PCP (RGB)	PCP (YC_BC_R)
čas[s]	35	3	4	753	99	53
max rozdíl(R)·10^3	2,1921	2,1044	2,1302	2,0346	2,0346	2,0592
max rozdíl(G)·10^3	2,0680	2,0172	2,0379	1,9967	1,9967	2,0131
max rozdíl(B)·10^3	2,4361	2,4154	2,4613	2,3573	2,3573	2,3856
prům.rozdíl(R)·10^3	2,1877	2,0995	2,1255	2,0346	2,0346	2,0592
prům.rozdíl(G)·10^3	2,0672	2,0161	2,0362	1,9967	1,9967	2,0131
prům.rozdíl(B)·10^3	2,4327	2,4136	2,4531	2,3573	2,3573	2,3856
min SSIM(R)	0,9747	0,9756	0,9706	0,9771	0,9771	0,9721
min SSIM(G)	0,9746	0,9755	0,9714	0,9775	0,9775	0,9734
min SSIM(B)	0,9560	0,9580	0,9514	0,9600	0,9600	0,9537
prům. SSIM(R)	0,9747	0,9756	0,9706	0,9771	0,9771	0,9721
prům. SSIM(G)	0,9746	0,9755	0,9714	0,9775	0,9775	0,9734
prům. SSIM(B)	0,9560	0,9580	0,9514	0,9600	0,9600	0,9537

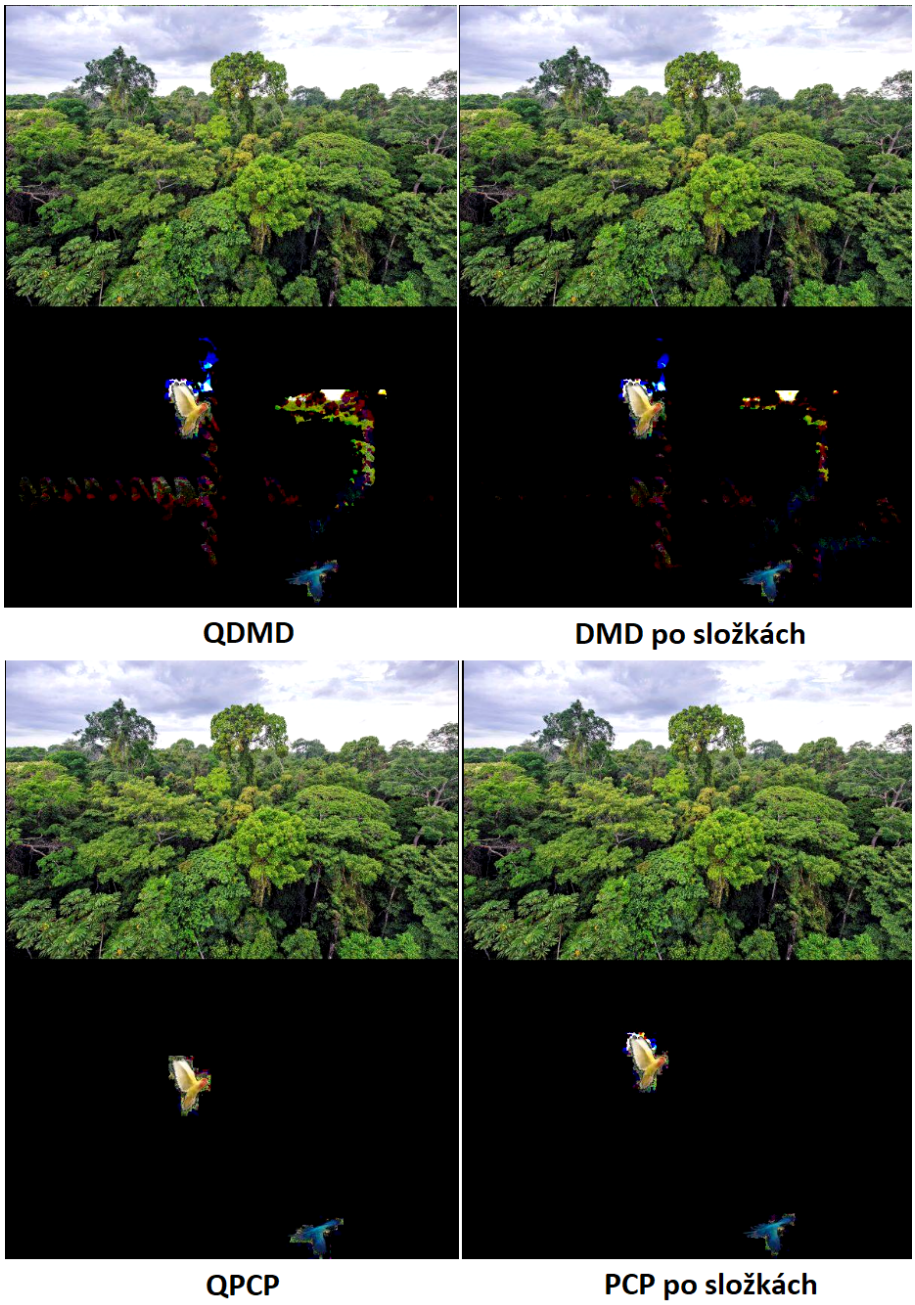
5.4 Výsledky na reálných datech

5.4.1 Video včel

Prvně metody vyzkoušíme na reálném videu včel ve stupních šedi. Toto video[2] je k nalezení ve složce `original_data` pod názvem `bees.mp4`. Na obrázku 5.8 je vyobrazen jeden snímek z tohoto videa. U tohoto videa můžeme předpokládat, že pozadí se v průběhu času nemění – předpokládáme rank $\mathbf{L} = 1$. Všechny dále zmíněné výsledky jsou k nalezení ve složce `bees`. Pro výsledky bude použito pouze škálované vykreslení. Filtrované vykreslení není pro tento typ videa vhodné, proto bude vynecháno.

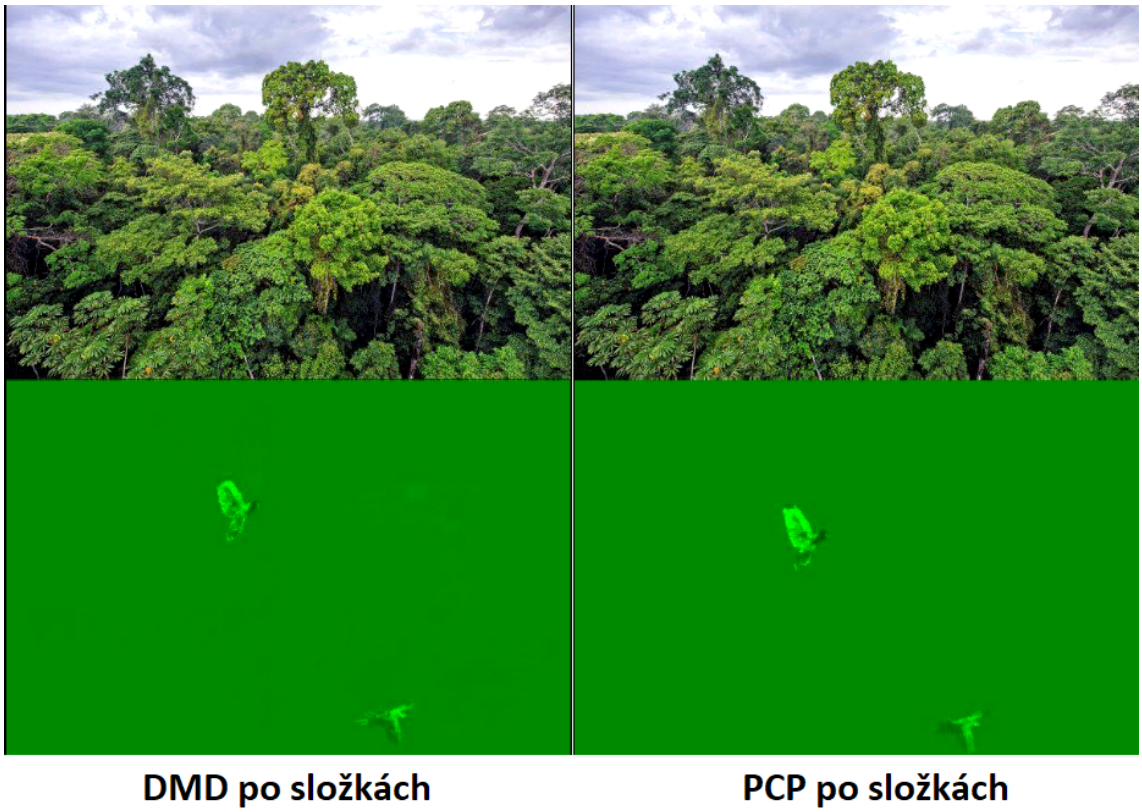
Nejprve budou k separaci použity konvexní metody. Výsledek je vyobrazen na obrázku 5.9. Na tomto obrázku je zeleně vyznačena část květiny, která byla pomocí mediánového filtru přidána do pozadí. Červeně je zvýrazněna tato stejná květina, která byla užitím PCP metody vyhodnocena jako součást pozadí a podle jejího

[2]Dostupné z `https://www.pexels.com/video/bee-farm-855504/`



Obr. 5.6: Porovnání jednotlivých metod pro barevná RGB data na simulovaných datech. V horní polovině obrazů výsledků pro jednotlivé metody je vykresleno odseparované pozadí, v dolní polovině je vykreslena získaná dynamická složka. Obrazy dynamické složky jsou vykresleny pomocí barevného filtrovaného vykreslení. Metoda značená jako QDMD odpovídá kvaternionovému DMD a metoda značená QPCP odpovídá kvaternionovému PCP. DMD po složkách a PCP po složkách označuje provedení DMD a PCP metod zvlášť pro jednotlivé barevné složky.

pohybu se mění i obraz pozadí. Vizuálně přesnou separaci provedla pouze DMD



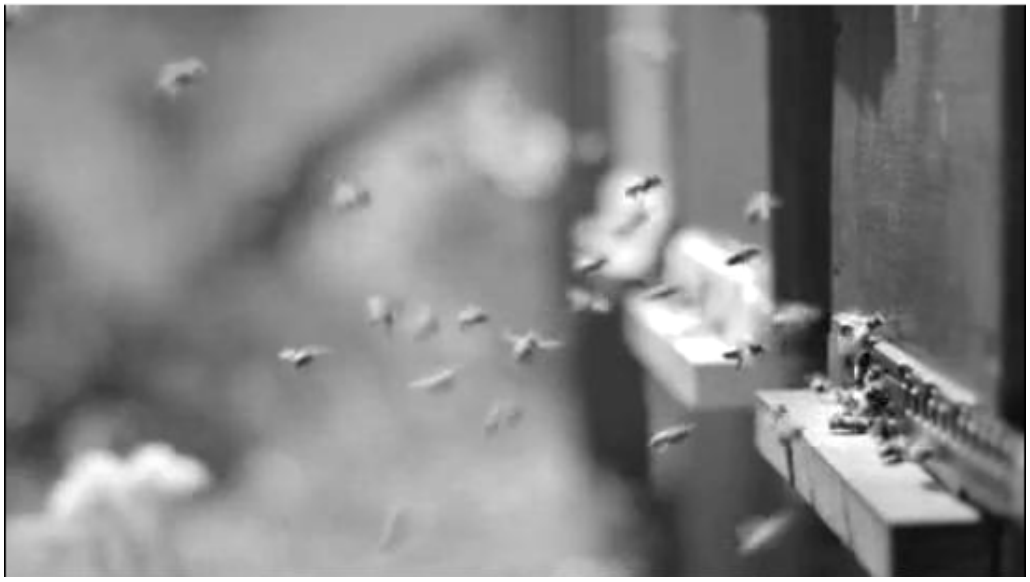
Obr. 5.7: Ukázka výsledků DMD a PCP metod provedených zvlášť pro jednotlivé barevné složky na simulovaných datech převedených na YC_BC_R. V horní polovině je vykresleno odseparované pozadí, v dolní polovině je vykreslena získaná dynamická složka.

metoda a konvexní RPCA metoda.

Časová náročnost, hodnost matice $\mathbf{L}$ a ℓ_1 norma matice $\mathbf{S}$ jsou pro jednotlivé metody vyobrazeny v tabulce 5.6. Z tabulky je patrné, že pouze mediánový filtr a DMD metoda získaly matici $\mathbf{L}$ s hodností jedna. Z tohoto důvodu a také kvůli velkému rozdílu v časové náročnosti DMD a konvexní RPCA metody volíme DMD metodu jako nejlepší metodu pro separaci tohoto videa.

Na videu včel byla vyzkoušena i metoda nekonvexního RPCA. Získat vhodné parametry λ a μ, které zajistí podobně dobrý nebo lepší výsledek než konvexní varianta $p = 1$, bylo velmi obtížné. Výsledná separace (pro různá p) byla i tak méně přesná než pro konvexní variantu nebo DMD. Navíc výpočet trval dvojnásobek času. Vzhledem k těmto výsledkům a faktu, že je těžké najít vhodné parametry, je nekonvexní RPCA shledáno jako nevhodné pro toto video.

Na video včel byly aplikovány i metody pro barevná data. Výsledky jsou k dispozici ve složce `bees\color`. Tyto výsledky byly stejné jako pro data ve stupních šedi. Tedy metody využívající PCP měnily výsledné pozadí podle pohybu květiny



Obr. 5.8: Snímek z videa včel.

Tab. 5.6: Porovnání výsledků konvexních metod na videu včel. Výsledky jsou porovnány pomocí časové náročnosti, výpočtem hodnosti získané matice pozadí $\mathbf{L}$ a pomocí ℓ_1 normy matice dynamické složky $\mathbf{S}$. Metoda PCP zkonvergovala po 167 iteracích a konvexní RPCA po 34 iteracích.

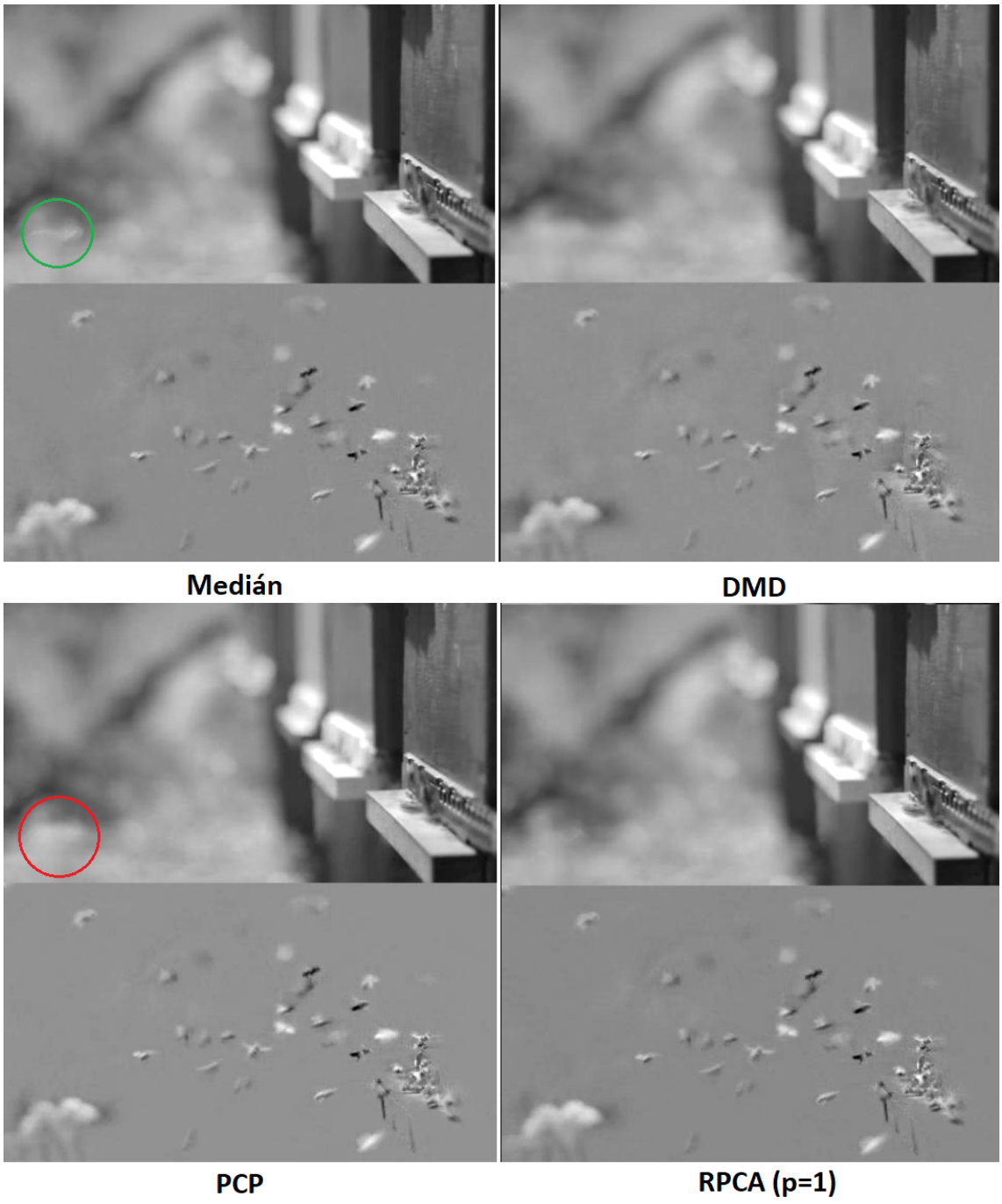
Výsledky konvexních metod na videu včel.			
	čas$[s]$	rank$(\mathbf{L})$	$\|\mathbf{S}\|_1 \cdot 10^7$
Mediánový filtr	0,3	1	12,829
DMD	1,7	1	14,511
PCP	380,9	93	9,154
RPCA s $p = 1$	69,5	24	10,908

ve větru, zatímco metody využívající DMD tuto květinu přenesly čistě do dynamické složky a pozadí zůstalo statické. Výsledky kvaternionového DMD a DMD po jednotlivých RGB složkách byly velmi podobné. DMD po složkách mělo menší hodnotu ℓ_1 normy matice $\mathbf{S}$ pro jednotlivé složky a také bylo značně rychlejší (viz tabulka 5.7). Tedy DMD po složkách se jeví jako nejlepší možná volba pro toto barevné video.

5.4.2 První video Slunce

Dalšími reálnými daty, na kterých budeme metody testovat, je video[3] erupce na Slunci poskytnuté prof. Druckmüllerem. Jednotlivé obrazy byly z důvodu výpočetní nároč-

[3]Dostupné z `http://www.zam.fme.vutbr.cz/~druck/SDO/Pm-nafe/2015_05_09/0-info.htm`



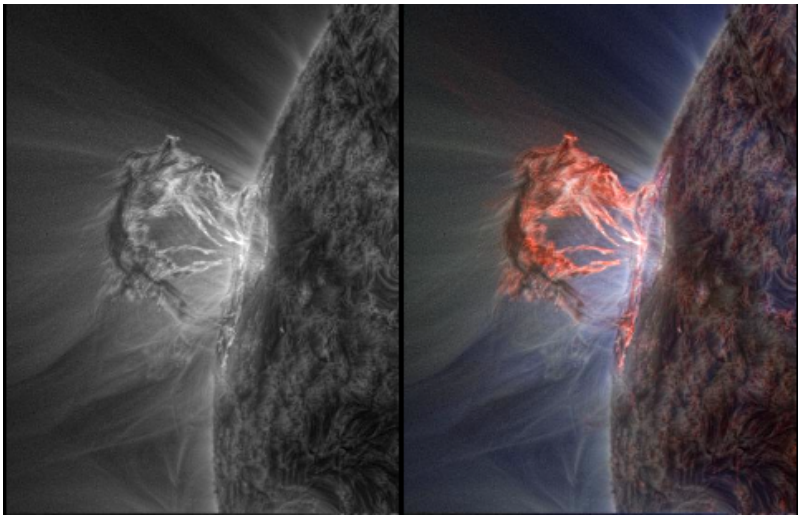
Obr. 5.9: Porovnání výsledků konvexních metod na videu včel. V horní polovině je vykresleno pozadí získané pomocí jednotlivých metod a v dolní polovině je vykreslena dynamická složka. Pro její vykreslení bylo použito škálované vykreslení. Zelený kroužek u metody mediánového filtru zvýrazňuje část pohybující se květiny, která byla přenesena do pozadí. Červený kroužek u PCP metody zvýrazňuje část pozadí, která se na každém snímku mění podle pohybu květiny.

Tab. 5.7: Porovnání výsledků DMD metod pro barevná data na videu včel. Výsledky jsou porovnány pomocí časové náročnosti, výpočtem hodnosti získané matice pozadí $\mathbf{L}$ (hodnost je stejná pro všechny barevné složky) a pomocí ℓ_1 normy matice dynamické složky $\mathbf{S}$.

Výsledky DMD metod pro barevná data na videu včel.					
	čas$[s]$	rank($\mathbf{L}$)	$\|\mathbf{S}^{\mathrm{R}}\|_1 \cdot 10^8$	$\|\mathbf{S}^{\mathrm{G}}\|_1 \cdot 10^8$	$\|\mathbf{S}^{\mathrm{B}}\|_1 \cdot 10^8$
QDMD	62	2	1,5479	1,5268	1,5543
DMD (RGB)	6	1	1,5067	1,4965	1,5122

nosti ořízнuty (viz obrázek 5.10) a zmenšeny na konečný počet 750 snímků o velikosti 420×545 px. Tyto snímky se nachází ve složce `flare_prom`.

Pro toto video předpokládáme, že pozadí netvoří pouze jediný obraz. Viditelný povrch Slunce se erupcí změní – změní se tedy i pozadí. Nejprve se opět zaměříme na data ve stupních šedi. Výsledky jsou uloženy ve složce `flare_prom\grayscale`.



Obr. 5.10: Snímky z prvního videa Slunce. Vpravo je vykreslen originální barevný snímek. Vlevo je daný snímek převeden do stupňů šedi.

Nejdříve byly vyzkoušeny konvexní metody separace. Časová náročnost výpočtu, hodnost matice $\mathbf{L}$ a ℓ_1 norma matice $\mathbf{S}$ získané pomocí jednotlivých konvexních metod jsou zapsány v tabulce 5.8. Z těchto výsledků jsme schopni metody porovnat pouze podle časové náročnosti. Přesnost separace nelze z této tabulky vyčíst, protože víme, že v tomto případě je očekávaná hodnost $\mathbf{L} > 1$. Jaká by měla být ideální získaná hodnost $\mathbf{L}$, lze pouze odhadnout.

Z tohoto důvodu bude vhodnější grafické srovnání výsledků jednotlivých konvexních metod. Toto srovnání je na obrázku 5.11, kde je vykreslen jeden snímek zís-

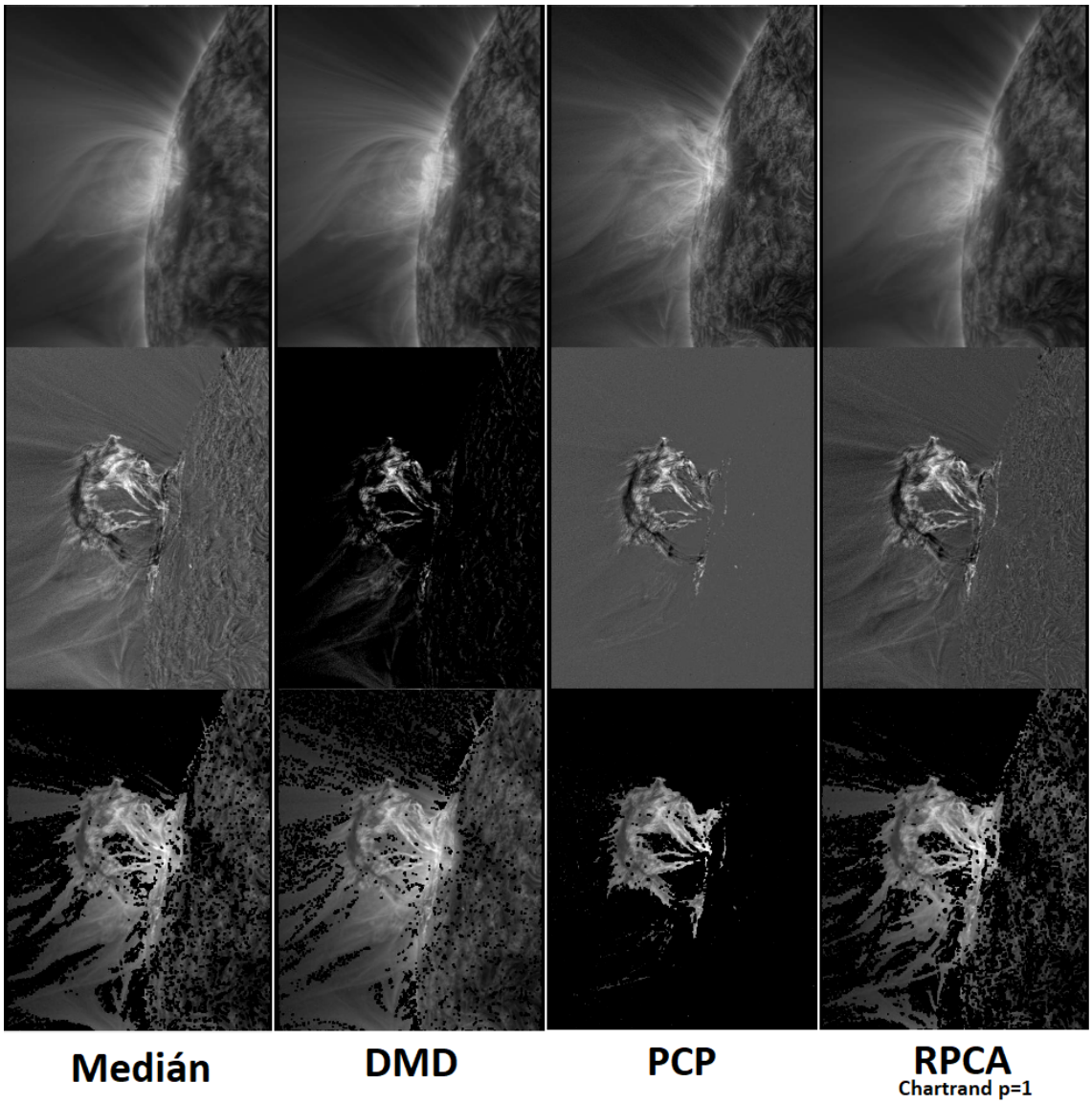
Tab. 5.8: Porovnání výsledků konvexních metod na prvním videu Slunce. Výsledky jsou porovnány pomocí časové náročnosti, výpočtem hodnosti získané matice pozadí $\mathbf{L}$ a pomocí ℓ_1 normy matice dynamické složky $\mathbf{S}$. Metoda PCP zkonvergovala po 100 iteracích a konvexní RPCA po 33 iteracích.

Výsledky konvexních metod na prvním videu Slunce.			
	čas$[s]$	rank$(\mathbf{L})$	$\|\mathbf{S}\|_1 \cdot 10^9$
Mediánový filtr	3	1	1,1202
DMD	168	3	1,2045
PCP	2544	332	0,4529
RPCA s $p = 1$	760	6	0,8402

kaného pozadí a dynamické složky pro každou konvexní metodu. Dynamická složka je vykreslena škálovaně i filtrovaně. Filtrované vykreslení odhaluje, jak velká část obrazu pozadí byla přeneseno do dynamické složky. Část pozadí byla přenesena do dynamické složky, protože povrch Slunce na tomto videu „šumí“. Může tedy působit jako pohybující se složka.

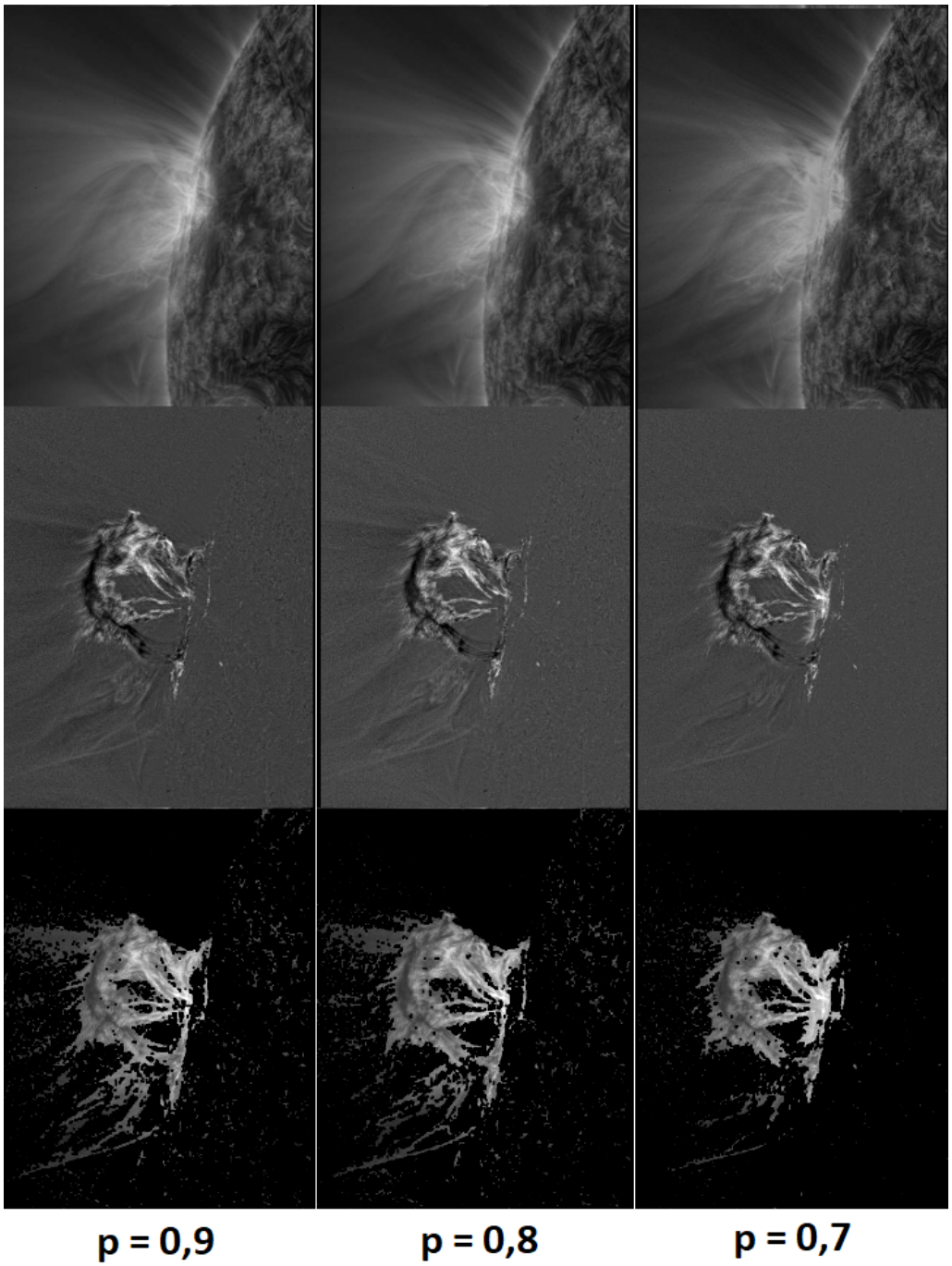
Z obrázku 5.11 je patrné, že jediná konvexní metoda, která nepřenesla obraz pozadí do dynamické složky, je PCP metoda. U mediánového filtru jsou očekávatelné horší výsledky, protože hodnost matice $\mathbf{L}$ bude při užití této metody vždy rovna jedné a jak již bylo řečeno, nejedná se o ideální výsledek. Zdá se, že konvexní RPCA dosáhla po PCP druhého nejlepšího výsledku. Proto je vhodné vyzkoušet, jak si povedou nekonvexní verze této metody.

Výsledky nekonvexního RPCA jsou shrnuty v tabulce 5.9. Z tabulky je patrné, že metoda pro $p = 0,7$ a pro $p = 0,6$ již získala matici $\mathbf{L}$ s vyšší hodností. Je tedy vhodné dané výsledky opět zkontrolovat vizuálně. Výsledky jsou vyobrazeny postupně na obrázcích 5.12, 5.13 a 5.14. Z těchto obrázků je zřejmé, že čím blíže je p k hodnotě $0,7$, tím přesnější je výsledek. Dále je zřejmé, že pro p blížící se k hodnotě nula je pozadí získané RPCA metodou téměř celý černý obraz. Celý původní obraz přejde do dynamické složky. Tedy výsledky pro $p \leq 0,5$ jsou velmi špatné. Pro $p = 0,7$ je výsledek srovnatelný s výsledkem PCP metody a jelikož byl získán třikrát rychleji, dá se považovat dokonce za lepší. Pro $p = 0,6$ lze metodou odseparovanou dynamickou složku považovat ze celou plochu, kde se na videu dějí pohyby. Tyto pohyby (převážně na povrchu Slunce) jsou velmi malé, ale při bližším zkoumání původního videa jsou přesto viditelné. Pro volbu nejlepší metody pro toto video tedy záleží, jestli chceme odseparovat pouze erupci, pak je nejvhodnější nekonvexní RPCA s $p = 0,7$ nebo veškeré pohyby, poté je nejvhodnější nekonvexní RPCA s $p = 0,6$.

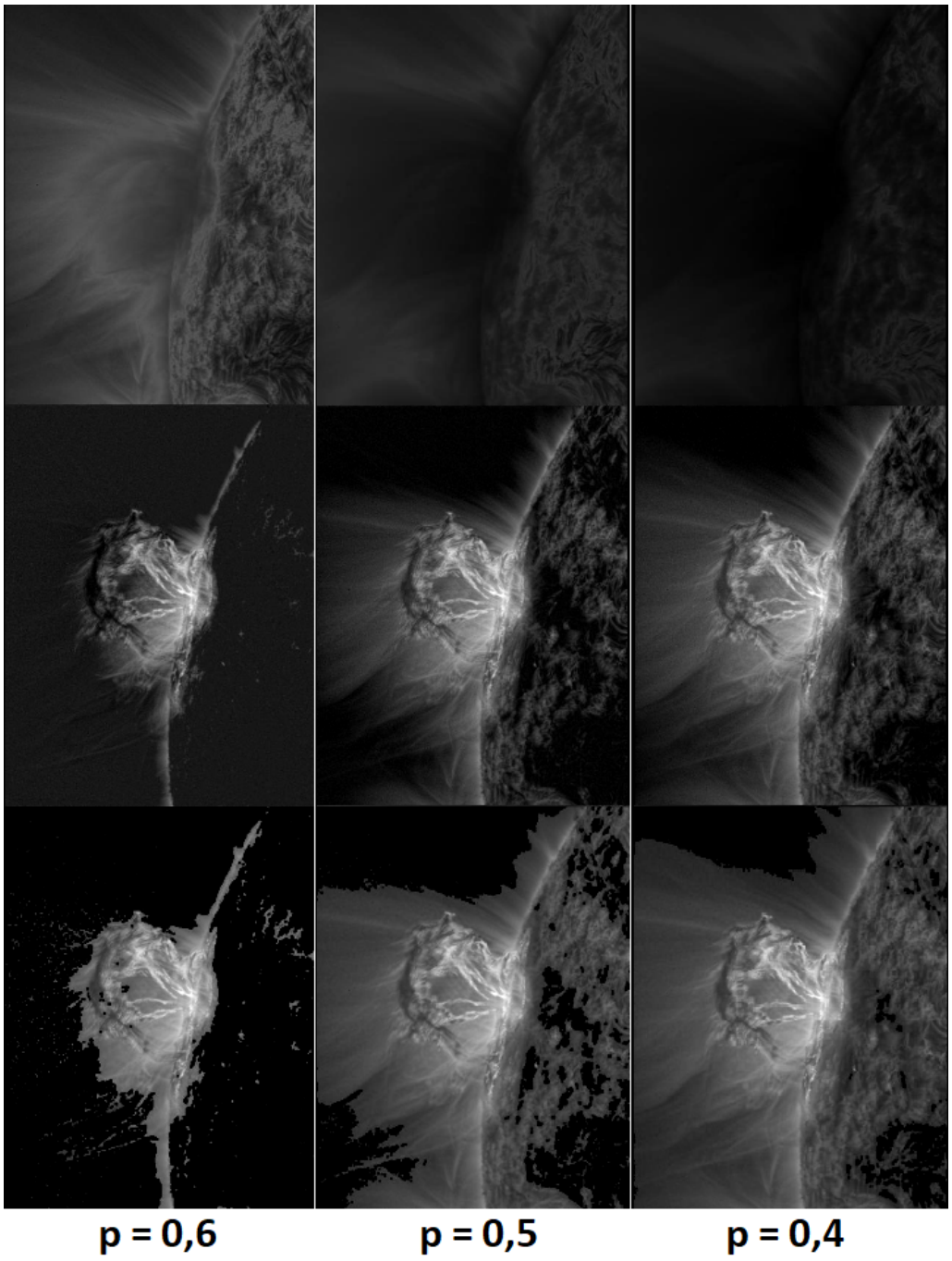


Obr. 5.11: Výsledek jednotlivých konvexních metod pro data ve stupních šedi získaný z prvního videa Slunce. V horní třetině je vykresleno odseparované pozadí, v druhé třetině je zobrazena dynamická složka vykreslená škálovaně a v dolní třetině je dynamická složka vykreslena pomocí Wienerova filtru a morfologického uzavření. Toto filtrované vykreslení především díky morfologickému uzavření zobrazuje dynamickou složku téměř jako původní obraz. To je způsobeno velkým množstvím šumu, který byl především mediánovým filtrem a DMD metodou přidán do dynamické složky. Je zřetelné, že mezi konvexními metodami má klasická PCP metoda nejlepší výsledky.

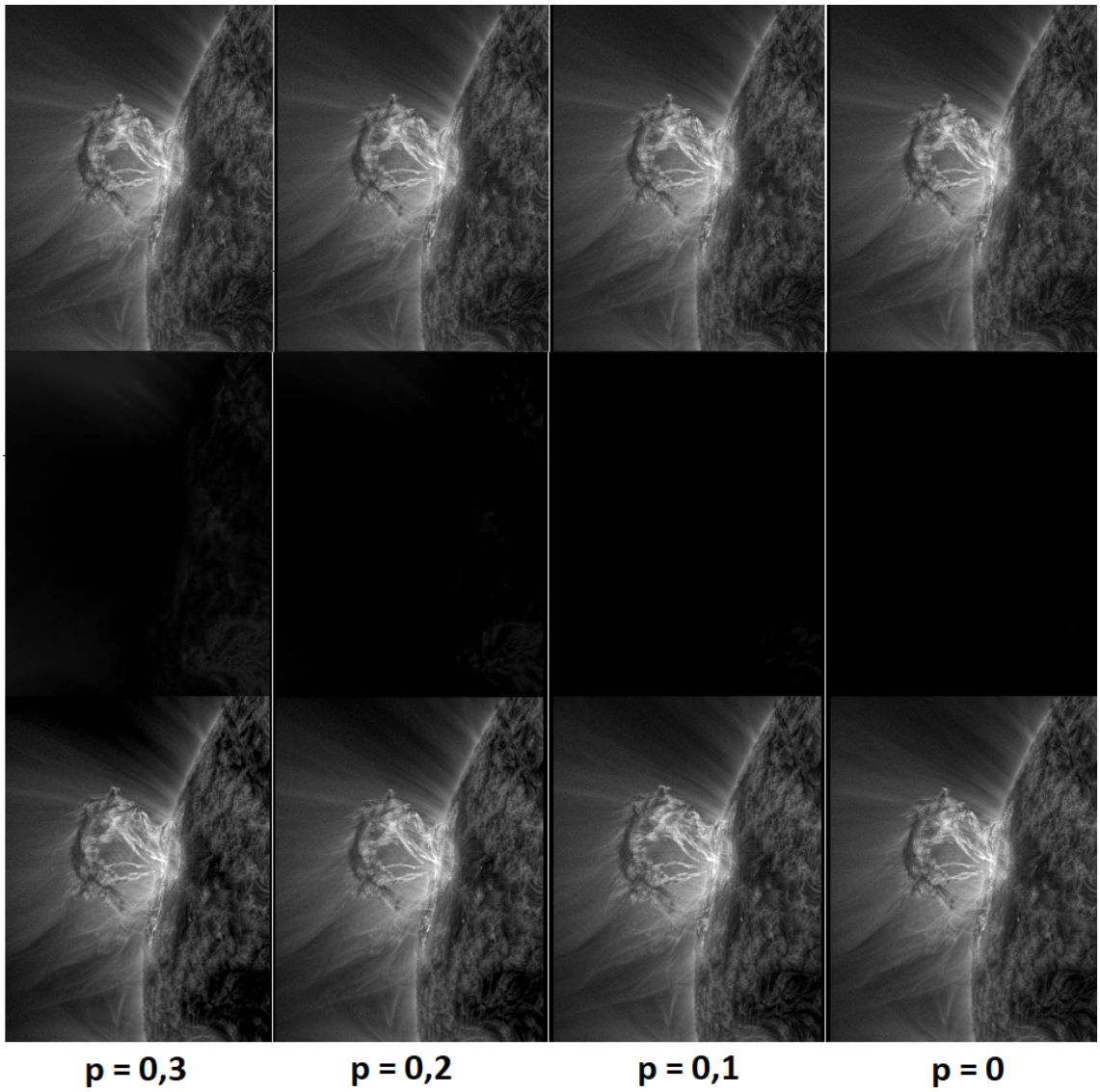
Dále se zaměříme na barevnou verzi tohoto videa Slunce. Výpočet z důvodu nedostatečné velikosti paměti RAM probíhal na počítači s procesorem Intel Xeon CPU E7-4820, 2GHz a 128GB RAM. Výsledky získané pomocí metod pro barevná data jsou dostupné ve složce `flare_prom\color`.



Obr. 5.12: Výsledek nekonvexní RPCA metody s $p = 0,9$, $p = 0,8$ a $p = 0,7$ pro první video Slunce. V horní třetině je vykresleno odseparované pozadí, v druhé třetině je zobrazena dynamická složka vykreslená škálovaně a v dolní třetině je dynamická složka vykreslena pomocí Wienerova filtru a morfologického uzavření. Je zřetelné, že metoda s $p = 0,7$ má nejlepší výsledek. Tento výsledek je vizuálně srovnatelný s výsledkem PCP metody.



Obr. 5.13: Výsledek nekonvexní RPCA metody s $p = 0,6$, $p = 0,5$ a $p = 0,4$ pro první video Slunce. V horní třetině je vykresleno odseparované pozadí, v druhé třetině je zobrazena dynamická složka vykreslená škálovaně a v dolní třetině je dynamická složka vykreslena pomocí Wienerova filtru a morfologického uzavření. Je zřetelné, že metoda s $p = 0,6$ má nejlepší výsledek. Je diskutabilní, jestli je tento výsledek lepší nebo horší než výsledek nekonvexního RPCA s $p = 0,7$ a výsledek PCP metody. Pro $p \leq 0,5$ je pozadí čím dál bližší černému obrazu a dynamická složka celému původnímu obrazu.



Obr. 5.14: Výsledek nekonvexní RPCA metody s $p \leq 0,3$ pro první video Slunce. V horní třetině je vykreslen původní obraz Slunce, v druhé třetině je vykresleno odseparované pozadí a v dolní třetí třetině je dynamická složka vykreslená škálovaně. Je zřetelné, že čím je p menší, tím je obraz pozadí bližší černému obrazu a dynamická složka celému původnímu obrazu.

Tab. 5.9: Výsledky nekonvexního RPCA pro různé hodnoty p na prvním videu Slunce. Výsledky jsou porovnány časovou náročností, výpočtem hodnosti získané matice pozadí $\mathbf{L}$ a pomocí ℓ_1 normy matice dynamické složky $\mathbf{S}$. Parametry λ a μ_0 byly nalezeny experimentálně. Parametr μ_0 (počátečního hodnota μ) je udáván v násobcích největšího singulárního čísla $\sigma_{\max}$ matice vstupních dat $\mathbf{M}$.

Výsledky nekonvexního RPCA na prvním videu Slunce.						
	λ	$\mu_0 \cdot \sigma_{\max}$	čas$[s]$	poč. iterací	rank($\mathbf{L}$)	$\|\mathbf{S}\|_1 \cdot 10^9$
0,9	0,0001	4/5	865	33	21	0,6651
0,8	0,00025	4/5	829	31	18	0,6673
0,7	0,00025	4/5	827	32	55	0,6190
0,6	0,00018	4/5	848	31	56	1,6667
0,5	0,00015	4/5	756	29	13	5,2545
0,4	0,00015	4/5	739	28	8	7,4560
0,3	0,00015	4/5	690	28	9	8,6101
0,2	0,00015	4/5	677	25	4	9,9861
0,1	0,00015	4/5	618	23	3	10,867
0	0,00015	4/5	441	20	3	11,422

Tab. 5.10: Výsledky metod pro barevná data na prvním videu Slunce. Výsledky jsou porovnány časovou náročností, výpočtem hodnosti získané matice pozadí $\mathbf{L}$ a pomocí ℓ_1 normy matice dynamické složky $\mathbf{S}$. Pro metody probíhající v $\mathrm{YC_BC_R}$ nepočítáme ℓ_1 normu matice $\mathbf{S}$ z důvodu nepřesností vzniklých převodem do RGB.

Výsledky metod pro barevná data na prvním videu Slunce.							
	čas $[min]$	rank($\mathbf{L}$) (R)	rank($\mathbf{L}$) (G)	rank($\mathbf{L}$) (B)	$\|\mathbf{S}\|_1 \cdot 10^9$ (R)	$\|\mathbf{S}\|_1 \cdot 10^9$ (G)	$\|\mathbf{S}\|_1 \cdot 10^9$ (B)
QDMD	25	6	6	6	1,6024	1,2222	1,2222
DMD(RGB)	12,5	3	3	3	1,5884	1,2657	1,2336
DMD($\mathrm{YC_BC_R}$)	12,6	3	3	3			
QPCP	805	750	750	750	0,0966	0,1329	0,1262
PCP(RGB)	255	113	101	90	0,4868	0,4584	0,4335
PCP($\mathrm{YC_BC_R}$)	209	78	70	81			

Časová náročnost, hodnost matice $\mathbf{L}$ a ℓ_1 norma matice $\mathbf{S}$ pro jednotlivé barevné složky jsou pro každou metodu rozepsány v tabulce 5.10. Z tabulky je patrné, že metody využívající PCP získaly matici $\mathbf{L}$ se značně vyšší hodností než metody

využívající DMD.

Výsledky metod můžeme opět porovnat vizuálně na obrázku 5.15. Dynamická složka je zde opět vykreslena škálovaným vykreslením i filtrovaným vykreslením. Z filtrovaného vykreslení je zřejmé, že metody pro barevná data využívající DMD mají stejný problém jako DMD metoda pro data ve stupních šedi – část pozadí je přenesena do dynamické složky. Porovnáme-li videa výsledků kvaternionového PCP, PCP po RGB složkách a PCP po YC_BC_R složkách, zjistíme, že kvaternionové PCP nechá část erupce v pozadí, tzn. pozadí se silně mění. PCP po složkách toto nedělá, proto je vyhodnoceno jako vhodnější. Vizuální rozdíl mezi PCP po složkách pro RGB a pro YC_BC_R data je téměř zanedbatelný. Pouze u YC_BC_R momentálně nejsme schopni správně vykreslit dynamickou složku. Proto pokud vyřešíme problém vykreslení, jsou oba přístupy pro toho video vhodné, protože dosahují nejlepších výsledků.

5.4.3 Druhé video Slunce

Dále bude testováno druhé video[4] Slunce, které zachycuje jiný typ sluneční erupce. Video bylo poskytnuto prof. Druckmüllerem. Jednotlivé obrazy byly opět z důvodu výpočetní náročnosti oříznuty (viz obrázek 5.16) a zmenšeny na konečný počet 999 snímků o velikosti 533×418 px. Tyto snímky se nachází ve složce `flare_5x4`.

Pro toto video opět předpokládáme, že pozadí netvoří pouze jediný obraz. Nejprve se opět zaměříme na data ve stupních šedi. Výsledky jsou uloženy ve složce `flare_5x4\grayscale`.

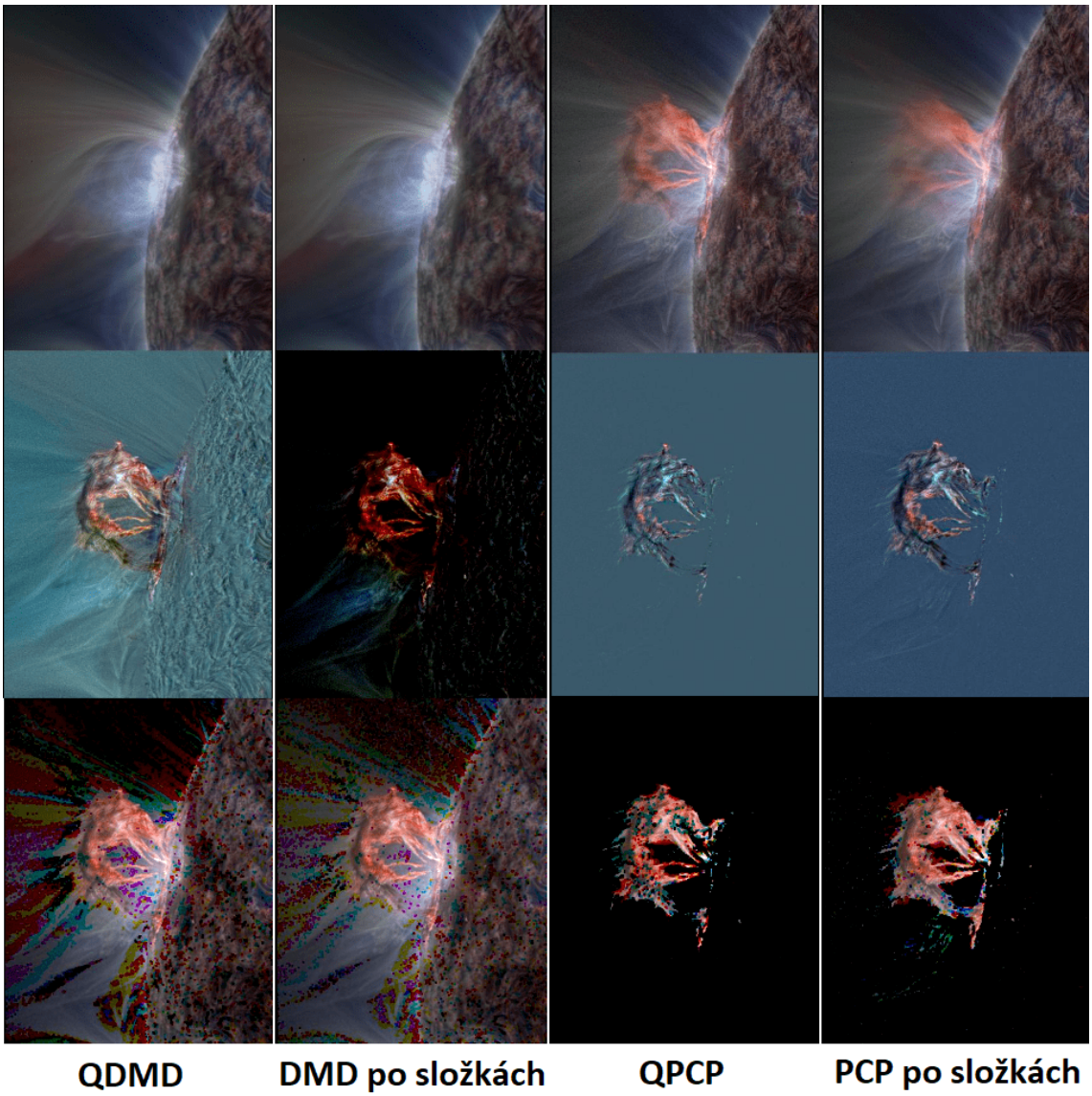
Prvně se zaměříme na konvexní metody. Srovnání výsledků je k dispozici v tabulce 5.11. Z tabulky je opět zřejmé, že pouze PCP a konvexní RPCA metoda získaly matici $\mathbf{L}$ s vyšší hodností. Na obrázku 5.17 jsou metody porovnány vizuálně.

Z obrázku 5.17 je patrné, že opět pouze PCP metoda nepřenesla pozadí do dynamické složky. Konvexní RPCA opět dosáhla druhého nejlepšího vizuálního výsledku. Vyzkoušíme tedy, jak si povedou nekonvexní varianty. Výsledky spolu s experimentálně nalezenými parametry λ a μ jsou k dispozici v tabulce 5.12.

Tentokrát z tabulky 5.12 nevidíme, že by hodnost matice $\mathbf{L}$ s menším p rostla a od určitého p opět klesala. Výsledky tedy zhodnotíme pouze vizuálně z obrázků 5.18 a 5.19. Výsledky pro $p \leq 0,3$ nejsou vykresleny, protože stejně jako v případě prvního videa Slunce RPCA s $p \leq 0,3$ získává obraz pozadí téměř celý černý a dynamická složka vypadá jako celý původní obraz.

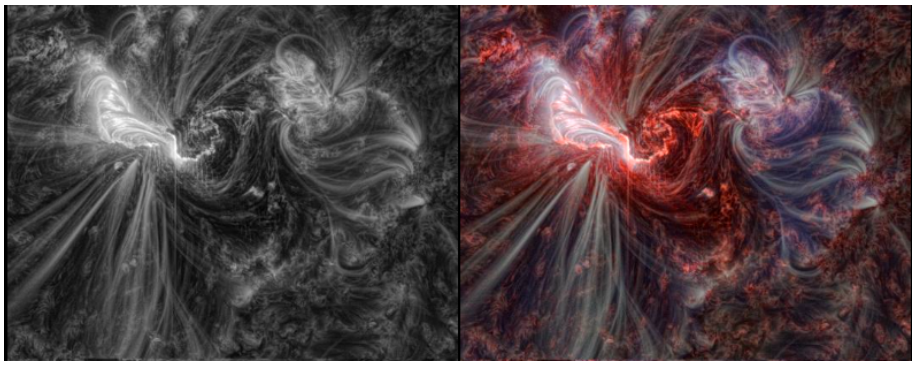
Z obrázků 5.18 a 5.19 je viditelné, že pro $p = 0,7$ a $p = 0,6$ má metoda opět nejlepší výsledky. Výsledky pro $p = 0,7$ jsou dokonce lepší než výsledky PCP metody. Opět lze říct, že nekonvexní RPCA s $p = 0,7$ skvěle odseparuje celý hlavní pohyb

[4]Dostupné z `http://www.zam.fme.vutbr.cz/~druck/SDO/Pm-nafe/2012_03_07/0-info.htm`

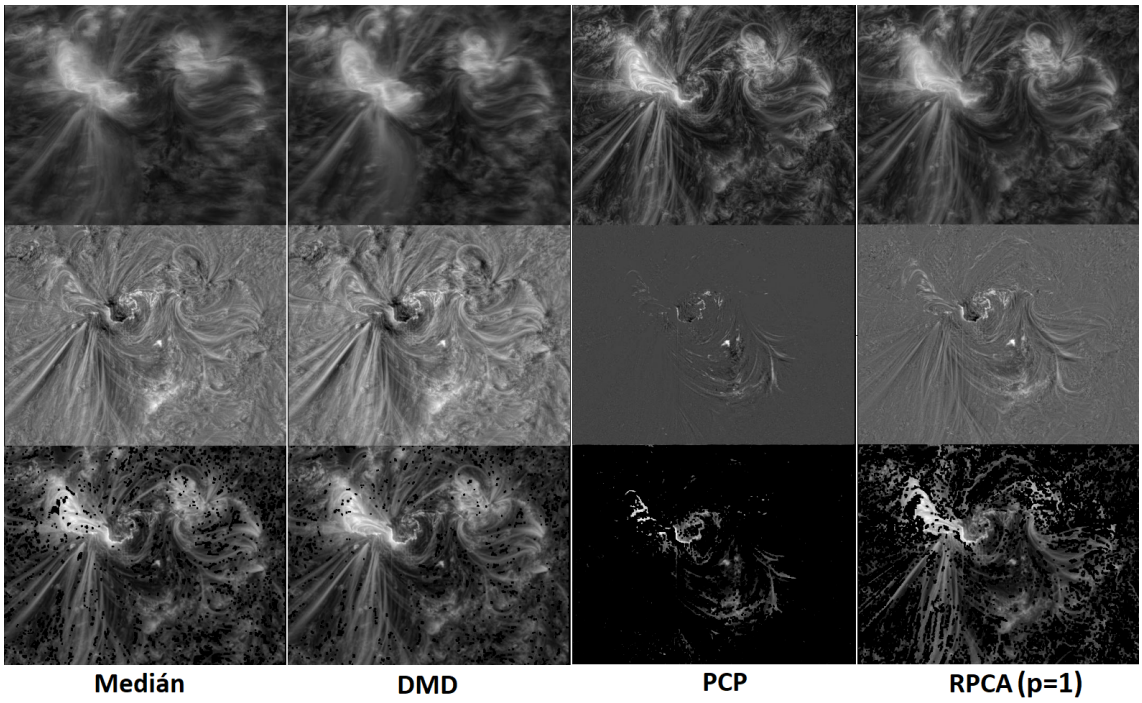


Obr. 5.15: Výsledek jednotlivých metod pro barevná data získaný z prvního videa Slunce. V horní třetině je vykresleno odseparované pozadí, v druhé třetině je zobrazena dynamická složka vykreslená škálovaně a v dolní třetině je dynamická složka vykreslena pomocí Wienerova filtru a morfologického uzavření. Toto filtrované vykreslení především díky morfologickému uzavření zobrazuje dynamickou složku téměř jako původní obraz. To je způsobeno velkým množstvím šumu, který byl především kvaternionovou DMD metodou a DMD metodou po složkách přidán do dynamické složky. Je zřetelné, že dané metody využívající PCP mají značně lepší výsledky než metody využívající DMD.

způsobený erupcí. Naproti tomu nekonvexní RPCA s $p = 0,6$ odseparuje veškerý pohyb na tomto videu. Tedy nekonvexní RPCA je vhodnou metodou pro toto video a volba p záleží na požadovaném výsledku.



Obr. 5.16: Snímky z druhého videa Slunce. Vpravo je vykreslen originální barevný snímek. Vlevo je daný snímek převeden do stupňů šedi.



Obr. 5.17: Výsledek jednotlivých konvexních metod pro data ve stupních šedi získaný z druhého videa Slunce. V horní třetině je vykresleno odseparované pozadí, v druhé třetině je zobrazena dynamická složka vykreslená škálovaně a v dolní třetině je dynamická složka vykreslena pomocí Wienerova filtru a morfologického uzavření. Toto filtrované vykreslení především díky morfologickému uzavření zobrazuje dynamickou složku téměř jako původní obraz. To je způsobeno velkým množstvím šumu, který byl především mediánovým filtrem a DMD metodou přidán do dynamické složky. Je zřetelné, že mezi konvexními metodami má klasická PCP metoda nejlepší výsledky.

Dále budou na tomto videu Slunce vyzkoušeny metody pro barevná data. Výpočet musel opět z důvodu nedostatku RAM paměti probíhat na počítači s proceso-

Tab. 5.11: Porovnání výsledků konvexních metod na druhém videu Slunce. Výsledky jsou porovnány pomocí časové náročnosti, výpočtem hodnosti získané matice pozadí $\mathbf{L}$ a pomocí ℓ_1 normy matice dynamické složky $\mathbf{S}$. Metoda PCP zkonvergovala po 72 iteracích a konvexní RPCA po 34 iteracích.

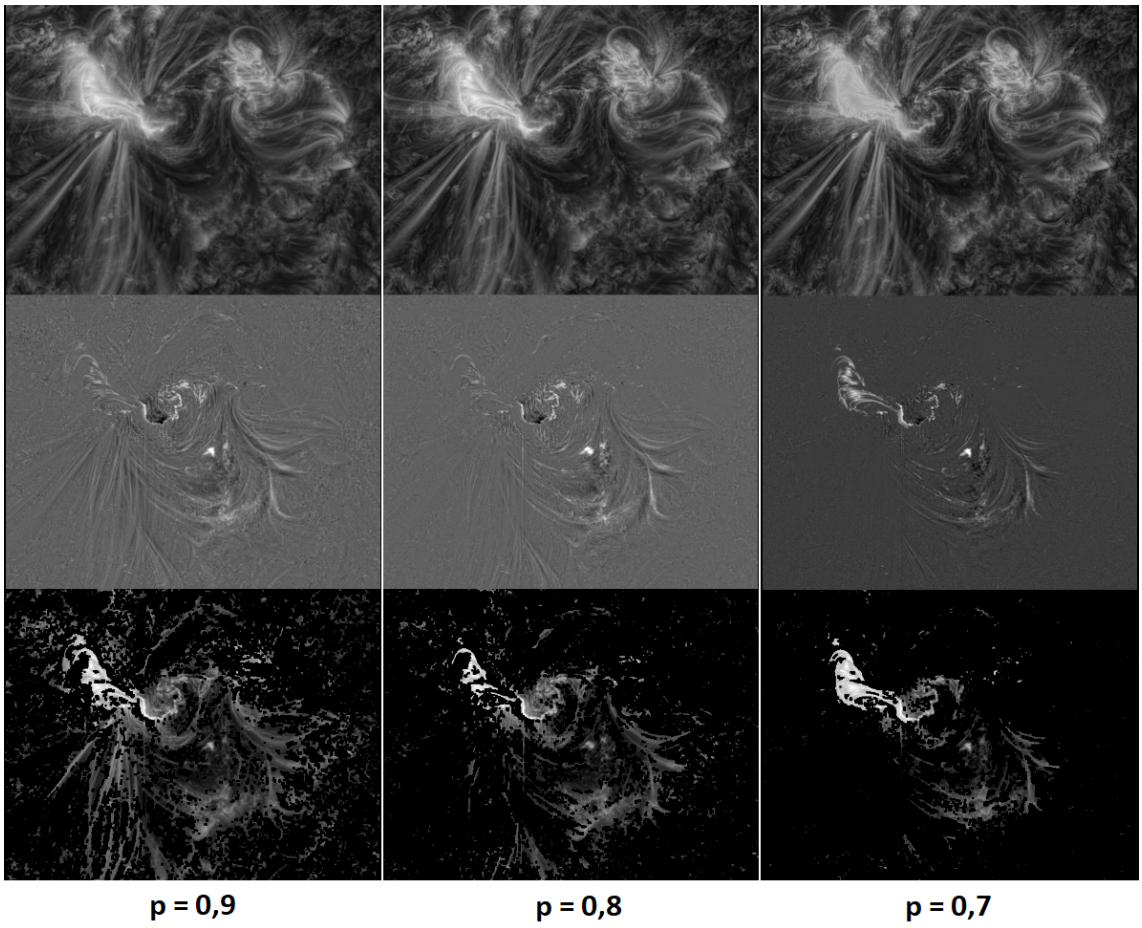
Výsledky konvexních metod na druhém videu Slunce.			
	čas$[s]$	rank($\mathbf{L}$)	$\|\mathbf{S}\|_1 \cdot 10^9$
Mediánový filtr	4	1	2,8965
DMD	258	3	3,1823
PCP	2410	506	3,1145
RPCA s $p = 1$	1404	17	1,2109

Tab. 5.12: Výsledky nekonvexního RPCA pro různé hodnoty p na druhém videu Slunce. Výsledky jsou porovnány časovou náročností, výpočtem hodnosti získané matice pozadí $\mathbf{L}$ a pomocí ℓ_1 normy matice dynamické složky $\mathbf{S}$. Parametry λ a μ_0 byly nalezeny experimentálně. Parametr μ_0 (počáteční hodnota μ) je udáván v násobcích největšího singulárního čísla $\sigma_{\max}$ matice vstupních dat $\mathbf{M}$.

Výsledky nekonvexního RPCA na druhém videu Slunce.						
	λ	$\mu_0 \cdot \sigma_{\max}$	čas$[s]$	poč. iterací	rank($\mathbf{L}$)	$\|\mathbf{S}\|_1 \cdot 10^9$
0,9	0,0003	4/5	1558	32	22	1,0066
0,8	0,00025	4/5	1437	32	50	0,7140
0,7	0,00025	4/5	1485	33	162	0,5046
0,6	0,00018	4/5	1522	33	208	1,9503
0,5	0,00015	4/5	1483	32	191	4,8916
0,4	0,00015	4/5	1521	33	298	7,6903
0,3	0,00015	4/5	1281	32	170	10,308
0,2	0,00015	4/5	1266	30	66	12,271
0,1	0,00015	4/5	1237	28	32	13,613
0	0,00015	4/5	1070	27	21	14,495

rem Intel Xeon CPU E7-4820, 2GHz a 128GB RAM. Získané výsledky jsou uloženy ve složce `flare_5x4\color`.

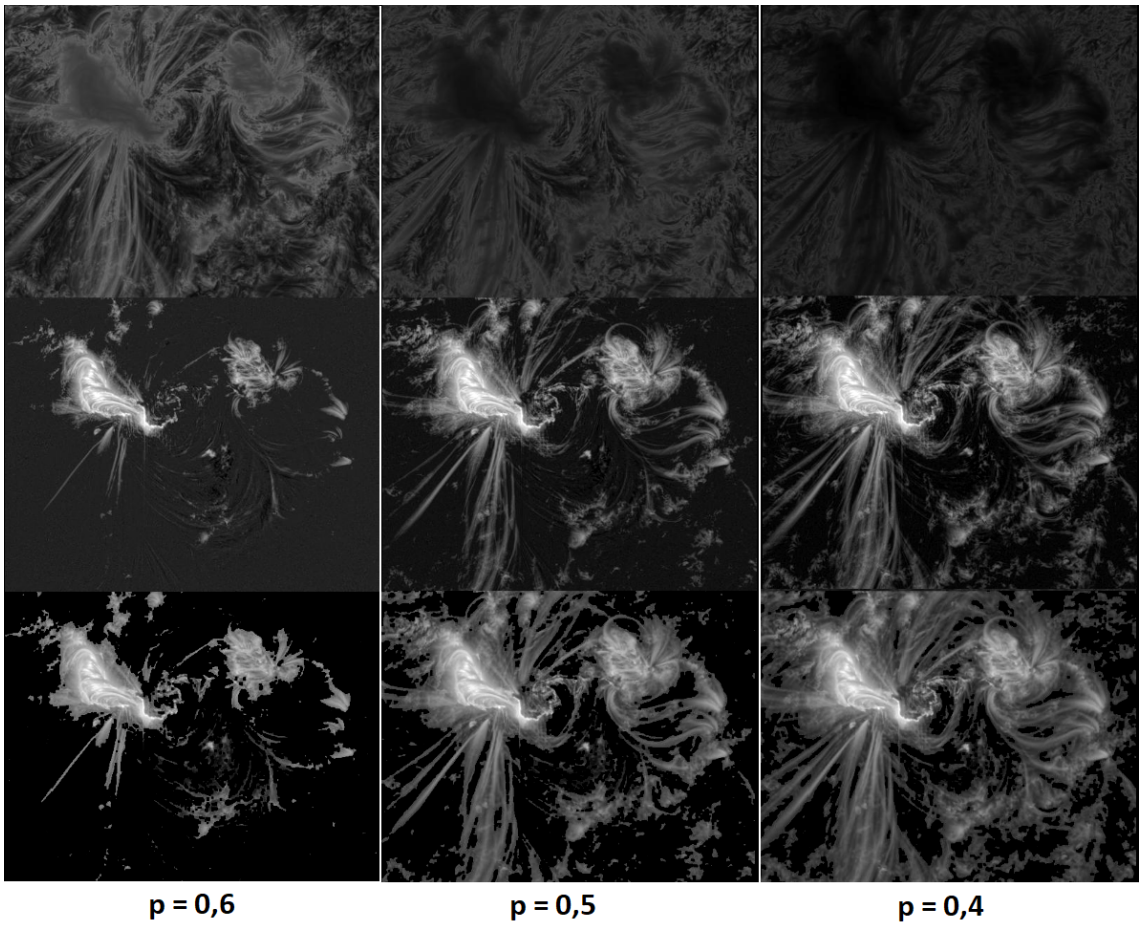
V tabulce 5.13 jsou metody porovnány podle časové náročnosti, hodnosti $\mathbf{L}$ a ℓ_1 normy $\mathbf{S}$ pro jednotlivé barevné složky. Z tabulky lze poznat, že stejně jako v případě prvního videa Slunce jsou výsledky metod využívajících DMD podobné jako pro data ve stupních šedi – pozadí bylo přeneseno i do dynamické složky. Tento fakt ověříme



Obr. 5.18: Výsledek nekonvexní RPCA metody s $p = 0,9$, $p = 0,8$ a $p = 0,7$ pro druhé video Slunce. V horní třetině je vykresleno odseparované pozadí, v druhé třetině je zobrazena dynamická složka vykreslená škálovaně a v dolní třetině je dynamická složka vykreslena pomocí Wienerova filtru a morfologického uzavření. Je zřetelné, že metoda s $p = 0,7$ má nejlepší výsledek. Tento výsledek se zdá být vizuálně dokonce lepší než výsledek PCP metody.

pomocí obrázku 5.20.

Na obrázku 5.20 byla vykreslena dynamická složka pouze škálovaně, protože filtrované vykreslení ji v případě metod užívajících DMD vykreslí téměř stejnou jako původní obraz. Je tedy zřejmé, že metody užívající PCP mají lepší výsledky. Rozdíl mezi výsledky kvaternionového PCP a PCP po RGB složkách je ukázán na obrázku 5.21. Při QPCP byly změny na Slunci opět více promítnuty do pozadí, než tomu bylo při užití PCP po složkách. Tomu odpovídají i získané hodnosti matice $\mathbf{L}$ pro jednotlivé barevné složky. Vizuálně je tedy zhodnoceno, že PCP po jednotlivých RGB složkách a PCP po jednotlivých $\mathrm{YC_BC_R}$ složkách dosahují pro toto video nejlepších výsledků.



Obr. 5.19: Výsledek nekonvexní RPCA metody s $p = 0,6$, $p = 0,5$ a $p = 0,4$ pro druhé video Slunce. V horní třetině je vykresleno odseparované pozadí, v druhé třetině je zobrazena dynamická složka vykreslená škálovaně a v dolní třetině je dynamická složka vykreslena pomocí Wienerova filtru a morfologického uzavření. Je zřetelné, že metoda s $p = 0,6$ má nejlepší výsledek. Tento výsledek by se dal dokonce považovat za přesný obraz složek, které se pohybují. Pro $p \leq 0,5$ je pozadí čím dál bližší černému obrazu a dynamická složka celému původnímu obrazu.

5.5 Zhodnocení výsledků metod

Z výše zmíněných výsledků pro data ve stupních šedi lze usoudit, že mediánový filtr je vhodná metoda pro videa, která mají neměnné pozadí. Dále je potřeba, aby se dynamická složka nevyskytovala na podobném místě ve většině obrazů. Poté jako v případě videa včel může nastat, že bude tato dynamická složka přidána do pozadí. Tedy největší výhodou mediánového filtru je jeho rychlost výpočtu, ale tato metoda není funkční pro všechny typy videí. Pro videa, kde se pozadí mění (např. videa Slunce), je tato metoda nevhodná, protože dynamická složka vypadá téměř jako původní obraz.

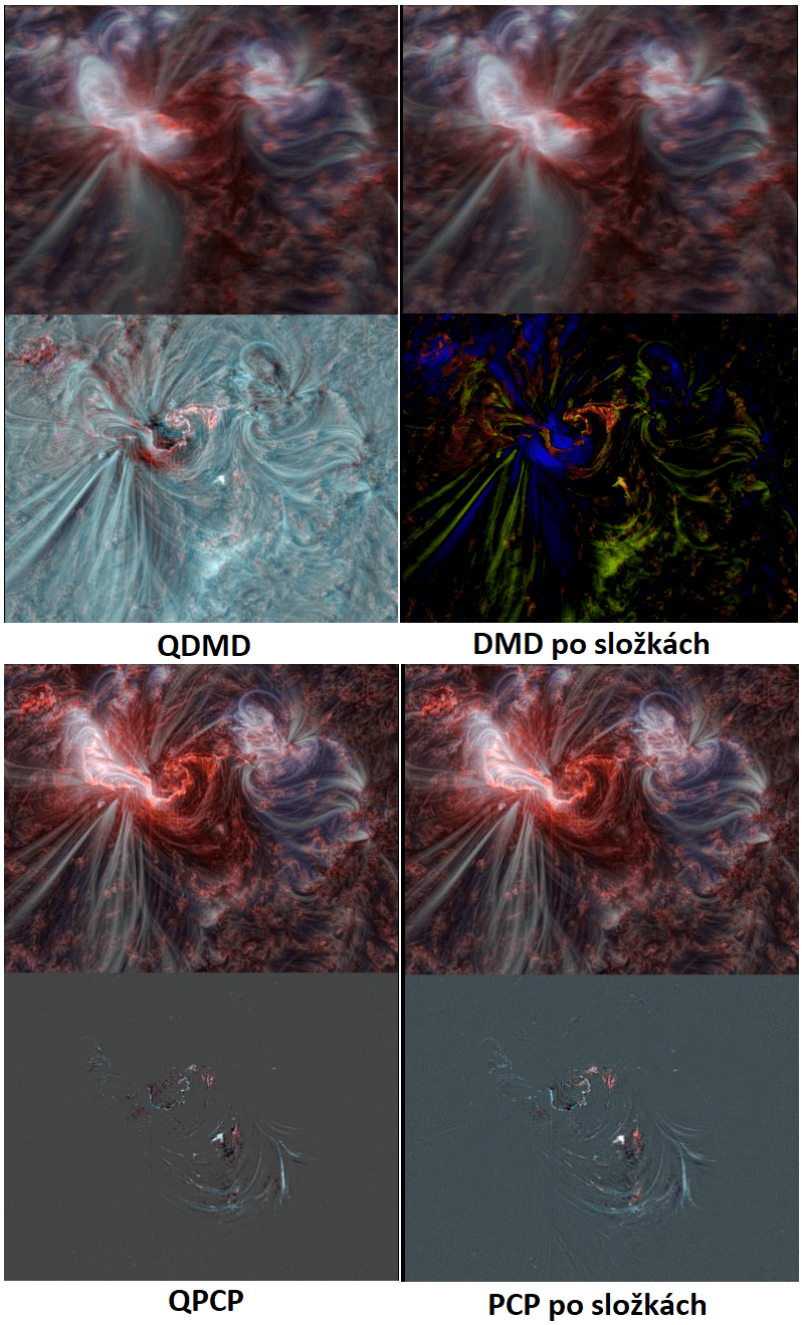
Tab. 5.13: Výsledky metod pro barevná data na druhém videu Slunce. Výsledky jsou porovnány časovou náročností, výpočtem hodnosti získané matice pozadí $\mathbf{L}$ a pomocí ℓ_1 normy matice dynamické složky $\mathbf{S}$. Pro metody probíhající v $\mathrm{YC_BC_R}$ nepočítáme ℓ_1 normu matice $\mathbf{S}$ z důvodů nepřesností vzniklých převodem do RGB.

Výsledky metod pro barevná data na druhém videu Slunce.							
	čas $[min]$	rank($\mathbf{L}$) (R)	rank($\mathbf{L}$) (G)	rank($\mathbf{L}$) (B)	$\|\mathbf{S}\|_1 \cdot 10^9$ (R)	$\|\mathbf{S}\|_1 \cdot 10^9$ (G)	$\|\mathbf{S}\|_1 \cdot 10^9$ (B)
QDMD	38	6	6	6	3,7674	3,2221	3,0973
DMD(RGB)	20	3	3	3	3,7164	3,2224	3,0479
DMD($\mathrm{YC_BC_R}$)	20	3	3	3			
QPCP	954	999	999	999	0,0660	0,1045	0,1009
PCP(RGB)	293	72	78	73	0,3386	0,3213	0,3135
PCP($\mathrm{YC_BC_R}$)	238	54	65	64			

Metoda DMD je také vhodná pro videa, která mají neměnné pozadí. Na rozdíl od mediánového filtru si dokáže poradit i s videi, kde se dynamická složka vyskytuje na podobném místě ve většině snímků. Jak bylo ukázáno na videích Slunce, tato metoda není vhodná pro videa, kterým se v průběhu času mění pozadí. Podobně jako pro mediánový filtr vypadá výsledná dynamická složka skoro jako původní obraz. Tato metoda je pořád relativně rychlá. Delší výpočetní náročnost než pro mediánový filtr je způsobena SVD rozkladem. Tento rozklad na rozdíl od PCP a RPCA probíhá pouze jednou. Tedy DMD metoda je druhá nejrychlejší.

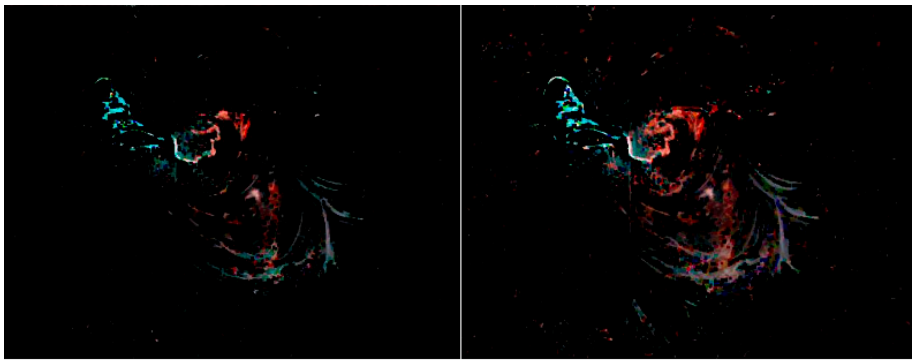
PCP metoda se ukazuje jako nejlepší pro získání přesných výsledků. Dokonce si dokáže velmi dobře poradit i s proměnným pozadím. Její hlavní nevýhodou je velká časová náročnost výpočtu, která je způsobena převážně výpočtem SVD rozkladu v každé iteraci. Jelikož PCP potřebovala ke konvergenci zpravidla alespoň dvakrát víc iterací než konvexní RPCA, je tato metoda vhodná především pro výpočty, které musí být přesné, ale nemusí být rychlé. Další případ, kdy PCP nemusí být vhodná metoda, je případ, kdy chceme získat neměnné pozadí. V takovém případě se může stát, že PCP metoda bude pozadí měnit v závislosti na pohybu pomalejší části dynamické složky (jako tomu bylo v případě květiny ve videu včel).

RPCA metoda s $p \in \langle 0, 1\rangle$ je vhodná zejména pro videa s proměnným pozadím. V těchto případech dosahuje dokonce lepších výsledků než PCP metoda. Navíc volbou p lze nastavovat, jestli v dynamické složce chceme pouze dominantní pohyby nebo veškeré pohyby, které ve videu nastávají. Obecně se ukazuje volba $p = 0,7$ a $p = 0,6$ jako nejlepší pro tyto případy. RPCA nemusí být vhodné pro videa se statickým pozadím, protože je velmi obtížné pro tyto typy videí najít vhodné para-



Obr. 5.20: Výsledek jednotlivých metod pro barevná data získaný z druhého videa Slunce. V horní polovině je vykresleno odseparované pozadí, v dolní polovině je zobrazena dynamická složka vykreslená škálovaně. Filtrované vykreslení bylo vynecháno, protože není vhodné pro výsledky metod využívajících DMD. Je zřetelné, že metody využívající PCP mají značně lepší výsledky než metody využívající DMD.

metry λ a μ. V případě videí Slunce s proměnným pozadím byla volba $\mu = 4/5 \cdot \sigma_{\max}$, kde $\sigma_{\max}$ je největší singulární číslo matice vstupních dat, vhodná pro všechna p. V případě videí s neměnným pozadím (jako byla simulovaná data a video včel), bylo



Obr. 5.21: Dynamická složka získaná z druhého videa Slunce pomocí metod využívajících PCP pro barevná data. Dynamická složka je vykreslena pomocí Wienerova filtru a morfologického uzavření.

třeba tento parametr měnit. Jelikož se může celá smyčka RPCA při špatné volbě parametrů zacyklit, je nalezení vhodných parametrů časově náročná záležitost. Navíc tato metoda stejně jako PCP musí v každé iteraci spočítat SVD rozklad, což ještě přidává na časové náročnosti. Ovšem v porovnání s PCP je počet potřebných iterací ke konvergenci značně menší. To je pravděpodobně způsobeno zmenšováním parametru μ v každé iteraci. Tedy RPCA s $p = 1$ může být za cenu menší přesnosti rychlejší náhradou PCP.

Dále z výše zmíněných výsledků pro barevná data lze usoudit, že stejně jako pro data ve stupních šedi jsou metody užívající DMD rozklad vhodné pro videa s neměnným pozadím. Jejich další výhodou je značně menší časová náročnost než časová náročnost metod využívajících PCP. Ovšem metody užívající DMD nejsou vhodné pro videa s proměnlivým pozadím, protože získají téměř statické pozadí a veškerý pohyb přiřadí do dynamické složky, která poté připomíná původní obraz.

Metody využívající PCP jsou naopak vhodnější pro videa s proměnným pozadím nebo také pro videa, kde je vyžadována přesnost, ale není vyžadována rychlost výpočtu. Kromě značné časové náročnosti nemusí být metody využívající PCP vhodné pro separaci videí, u kterých chceme získat neměnné pozadí. Je možné, že tyto metody budou měnit pozadí v závislosti na pohybu pomalejší části dynamické složky.

Kvaternionové metody – QPCP a QDMD dosahují podobně dobrých výsledků, jako když použijeme klasické PCP a DMD zvlášť na jednotlivé barevné složky. Navíc z důvodu výpočtu SVD rozkladu na čtyřikrát větší matici než je tomu u klasických metod, jsou velmi výpočetně náročné. Tedy usuzujeme, že výpočet PCP a DMD pro jednotlivé barevné složky zvlášť je dostatečný. Jediný případ, kdy by kvaternionové metody mohly být lepší, je takový, kdy požadujeme, aby pozadí bylo více proměnné. V takovém případě kvaternionové metody zpravidla získávají matici pozadí $\mathbf{L}$ s větší

hodností než metody klasické. To je zřejmě způsobeno provázaností jednotlivých barevných složek při výpočtu SVD rozkladu.

PCP a DMD metody pro jednotlivé barevné složky dat převedených do $\mathrm{YC_BC_R}$ dosáhly slušných výsledků. Dle získaných obrazů pozadí lze říct, že výsledky byly podobné jako při užití dat v RGB. Pro přesnější zhodnocení je potřeba vymyslet způsob, jak tyto data vykreslit, aby hodnota pixelu $[0, 0, 0]$ v $\mathrm{YC_BC_R}$ odpovídala pixelu s hodnotou $[0, 0, 0]$ v RGB.

Pokud pomocí výše popsaných metod nejsme schopni získat požadovaný vzhled dynamické složky, nabízí se vyzkoušet mrDMD metodu, která je schopna odseparovat pouze pomaleji se pohybující složky. Bohužel nevýhodou této metody je, že musí být silně optimalizována pro daná data. Když byla tato metoda vyzkoušena na videu Slunce a včel, nebylo dosaženo žádných rozumných výsledků. Pohyb získaný v jednotlivých dekompozičních úrovních většinou pouze odpovídal obrazu míst, kde se pomalejší dynamická složka nachází, ale tento pohyb nebyl zaznamenán.

Závěr

V této diplomové práci byly nastudovány metody rozkladu videosekvencí na více složek s různou dynamikou. Tato práce se zaměřila na čtyři metody separace statické a dynamické složky určené pro data ve stupních šedi – mediánový filtr, DMD metodu, PCP metodu a RPCA metodu, přičemž u RPCA metody bylo možné volit mezi konvexní a nekonvexní variantou. Dále byly tyto metody rozšířeny pro barevná data. Kromě intuitivního přístupu, kdy byly jednotlivé metody aplikovány na každou barevnou složku zvlášť a výsledek vznikl složením výsledků jednotlivých barevných složek, byly představeny i metody jako kvaternionové PCP a kvaternionové DMD, které zachovávají provázanost jednotlivých barevných složek. Bylo také vyzkoušeno převedení barevných dat z RGB do $\mathrm{YC_BC_R}$ a poté aplikování metod na jednotlivé složky. V neposlední řadě byla představena metoda Multiresolution DMD, která je schopna rozložit i dynamickou složku na více složek podle jejich rychlosti pohybu.

Výše zmíněné metody byly detailně vysvětleny a odvozeny. Dále byly tyto metody implementovány v Matlabu. Implementovány byly i funkce pro předzpracování dat, vykreslení a porovnání získaných výsledků. Následně byly představeny parametry, které je potřeba pro jednotlivé metody nastavit. Pro některé parametry byla nalezena hodnota fungující pro všechna vstupní data. Jiné parametry bylo potřeba určit experimentálně pro každá vstupní data. Poté byly metody vyzkoušeny na simulovaných i reálných datech. Jako reálná data byla volena videa s neměnným, ale i s částečně proměnlivým pozadím. Výsledky metod provedených na simulovaných datech byly poté porovnány s původním obrazem pozadí pomocí Frobeniovy normy a SSIM indexu. Výsledky na reálných datech mohly být zhodnoceny pouze vizuálně.

Otestováním metod na simulovaných a reálných datech se ukázalo, že pro videa s neměnným pozadím je vhodnější metoda mediánového filtru a DMD metoda. Pokud dané video navíc obsahuje dynamickou složku na podobném místě pro většinu snímků, je lepší volit DMD metodu. Obě tyto metody jsou poměrně rychlé a bylo by možné je použít i pro separaci v reálném čase, omezíme-li se např. pouze na průmyslové kamery s nižším rozlišením a 20–30 snímky za sekundu. Pokud máme video s proměnlivým pozadím, ukazuje se jako nejlepší použít nekonvexní RPCA s $p = 0,7$ nebo $p = 0,6$. Touto volbou je možné částečně kontrolovat, jestli bude do dynamické složky přiřazen pouze dominantní nebo veškerý pohyb. Pokud vyžadujeme pro obecné video velmi přesné výsledky a není vyžadováno, aby byl výpočet rychlý, je vhodné použít PCP metodu. Ta má v obecném případě nejpřesnější výsledky ale také největší výpočetní náročnost, která je způsobena SVD rozkladem. Pro barevná videa bylo ukázáno, že je dostatečné provést metody separace zvlášť pro jednotlivé barevné složky. Tento přístup dosahuje stejně dobrých, často i lepších výsledků než kvaternionové verze DMD a PCP. Stejně jako pro data ve stupních šedi i pro barevná

data je DMD rychlejší a vhodné pro videa s neměnným pozadím, zatímco PCP je výpočetně náročné a je vhodnější pro videa s proměnlivým pozadím. Pokud jsou data před aplikací metod separace na jednotlivé barevné složky převedena do YC_BC_R, výsledek vypadá srovnatelně s RGB prostorem. Pro přesné určení by bylo potřeba vymyslet lepší způsob vykreslení výsledků, které vznikly v YC_BC_R a pro zhodnocení byly převedeny zpět do RGB. Dále se ukázalo, že užití Multiresolution DMD metody není bez velké optimalizace vhodné pro všechna vstupní videa. Z některých videí jsme ale schopni získat i výsledky, které jsou okem těžko viditelné.

V průběhu práce byl nalezen nečekaný problém. Kvaternionové DMD implementované pro komplexní reprezentace kvaternionových matic nefunguje zcela správně. Tento problém je vhodný námět k dalšímu studiu.

Do budoucna by bylo vhodné zkusit zrychlit výpočet PCP, které má jinak velmi přesné výsledky. Toho by mohl docílit např. algoritmus z článku [29]. Také by bylo vhodné zaměřit se na optimalizaci Multiresolution DMD, aby bylo schopno získat rozumný výsledek pro všechna vstupní videa. V neposlední řadě by bylo vhodné vzhledem k současnému trendu zakomponovat do jednotlivých metod neuronové sítě. Tento přístup byl zvolen např. v článku [17], ve kterém jeho autoři spojili neuronové sítě a DMD metodu.

Literatura

[1] BAUSCHKE, Heinz H. a Patrick L. COMBETTES. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces.* Journal of Computational and Applied Mathematics [online]. 2017, 2011, 322(1), 109-128 [cit. 2023-05-05]. ISSN 03770427. Dostupné z: doi:10.1007/978-1-4419-9467-7

[2] BOUWMANS, Thierry, B. ROSARIO, A.P. PENTLAND a Chandrika KAMATH. *Subspace Learning for Background Modeling: A Survey.* Recent Patents on Computer Science [online]. 2010, 2004-1-7, 2(3), 223-234 [cit. 2023-04-28]. ISSN 18744796. Dostupné z: doi:10.2174/1874479610902030223

[3] BRUNTON, Steven L. a J. Nathan KUTZ. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.* Cambridge: Cambridge University Press, 2019. ISBN 978-1108422093.

[4] CANDÈS, Emmanuel J., Xiaodong LI, Yi MA a John WRIGHT. *Robust principal component analysis?* Journal of the ACM [online]. 2011, 58(3), 1-37 [cit. 2020-06-07]. DOI: 10.1145/1970392.1970395. ISSN 0004-5411. Dostupné z: https://dl.acm.org/doi/10.1145/1970392.1970395

[5] CHAN, Tak-Shing T. a YI-HSUAN YANG. *Complex and Quaternionic Principal Component Pursuit and Its Application to Audio Separation.* IEEE Signal Processing Letters [online]. 2016, 23(2), 287-291 [cit. 2023-05-12]. ISSN 1070-9908. Dostupné z: doi:10.1109/LSP.2016.2514845

[6] CHARTRAND, R. *Nonconvex Splitting for Regularized Low-Rank Sparse Decomposition.* IEEE Transactions on Signal Processing [online]. 2012, 60(11), 5810-5819 [cit. 2023-05-05]. ISSN 1053-587X. Dostupné z: doi:10.1109/TSP.2012.2208955

[7] CHEN, Kevin K., Jonathan H. TU a Clarence W. ROWLEY. *Variants of Dynamic Mode Decomposition: Boundary Condition, Koopman, and Fourier Analyses.* Journal of Nonlinear Science [online]. 2012, 22(6), 887-915 [cit. 2023-03-24]. ISSN 0938-8974. Dostupné z: doi:10.1007/s00332-012-9130-9

[8] CHEUNG, Sen-ching S., Sethuraman PANCHANATHAN, Bhaskaran VASUDEV and Chandrika KAMATH. *Robust techniques for background subtraction in urban traffic video* [online]. 2004-1-7, 881- [cit. 2023-04-28]. Dostupné z: doi:10.1117/12.526886

[9] DAŇKOVÁ, M. *Komprimované snímání v perfuzním zobrazování pomocí magnetické rezonance.* Diplomová práce, Vysoké učení technické v Brně, 2014

[10] GEBRTOVÁ, Karolína. *Separace dynamické a statické složky v sérii obrazů.* Brno, 2020, 63 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav matematiky. Vedoucí práce: doc. Mgr. Pavel Rajmic, Ph.D.

[11] GROSEK J., J. Nathan KUTZ. *Dynamic Mode Decomposition for Real-Time Background/Foreground Separation in Video* [online].University of Washington, 2014.[cit. 2020-06-09]. Dostupné z: https://arxiv.org/pdf/1404.7592.pdf

[12] GUYON, Charles, Thierry BOUWMANS a El-hadi ZAHZAH. *Robust Principal Component Analysis for Background Subtraction: Systematic Evaluation and Comparative Analysis.* Principal Component Analysis [online]. InTech, 2012, 2012-03-02 [cit. 2023-05-03]. ISBN 978-953-51-0195-6. Dostupné z: doi:10.5772/38267

[13] H. TU, Jonathan, Clarence W. ROWLEY, Dirk M. LUCHTENBURG, Steven L. BRUNTON a J. Nathan KUTZ. *On dynamic mode decomposition: Theory and applications.* Journal of Computational Dynamics [online]. 2014, 1(2), 391-421 [cit. 2023-03-25]. ISSN 2158-2505. Dostupné z: doi:10.3934/jcd.2014.1.391

[14] HAN, Juan, Kit Ian KOU a Jifei MIAO. *Quaternion-based dynamic mode decomposition for background modeling in color videos.* Computer Vision and Image Understanding [online]. 2022, 224 [cit. 2023-05-11]. ISSN 10773142. Dostupné z: doi:10.1016/j.cviu.2022.103560

[15] HUBER, Peter J. *Robust Estimation of a Location Parameter.* The Annals of Mathematical Statistics [online]. 1964, 35(1), 73-101 [cit. 2023-05-05]. ISSN 0003-4851. Dostupné z: doi:10.1214/aoms/1177703732

[16] GOLUB, Gene H. a Charles F. VAN LOAN. *Matrix Computations: Johns Hopkins Studies in the Mathematical Sciences : Book 3* [online]. 4th ed. Johns Hopkins University Press, 2013 [cit. 2023-05-13]. ISBN 9781421408590. Dostupné z: https://books.google.cz/books?id=5U-l8U3P-VUC

[17] KOBAYASHI, E., H. YASUDA, K. HAYASAKA, Y. OTAKE, S. ONO a S. *MURAMATSU. Multi-Resolution Convolutional Dictionary Learning for Riverbed Dynamics Modeling.* ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [online]. IEEE, 2023, 2023-6-4, 1-5 [cit. 2023-05-24]. ISBN 978-1-7281-6327-7. Dostupné z: doi:10.1109/ICASSP49357.2023.10096452

[18] KRISHNAN, Dilip a Rob FERGUS. *Fast Image Deconvolution using Hyper-Laplacian Priors.* Advances in Neural Information Processing Systems [online].

Curran Associates, 2009, 22(3) [cit. 2023-05-07]. ISSN 0730-0301. Dostupné z: https://proceedings.neurips.cc/paper_files/paper/2009/file/3dd48ab31d016ffcbf3314df2b3cb9ce-Paper.pdf

[19] KUTZ, Jose Nathan, Steven L. BRUNTON, Bingni W. BRUNTON a Joshua L. PROCTOR. *Dynamic mode decomposition: data-driven modeling of complex systems.* Philadelphia: SIAM, Society for Industrial and Applied Mathematics, [2016]. ISBN 978-1-61197-449-2.

[20] KUTZ, J. Nathan, Xing FU a Steven L. BRUNTON. *Multiresolution Dynamic Mode Decomposition.* SIAM Journal on Applied Dynamical Systems [online]. 2016, 15(2), 713-735 [cit. 2023-04-18]. ISSN 1536-0040. Dostupné z: doi:10.1137/15M1023543

[21] KUTZ, J. Nathan, Xing FU, Steve L. BRUNTON a N. Benjamin ERICHSON. *Multi-resolution Dynamic Mode Decomposition for Foreground/Background Separation and Object Tracking.* 2015 IEEE International Conference on Computer Vision Workshop (ICCVW) [online]. IEEE, 2015, 2015, 15(2), 921-929 [cit. 2023-04-18]. ISBN 978-1-4673-9711-7. ISSN 1536-0040. Dostupné z: doi:10.1109/ICCVW.2015.122

[22] LANGE, Kenneth. *Vector and Matrix Norms. Numerical Analysis for Statisticians* [online]. New York, NY: Springer New York, 2010, 2010-04-19, 77-91 [cit. 2023-05-13]. Statistics and Computing. ISBN 978-1-4419-5944-7. Dostupné z: doi:10.1007/978-1-4419-5945-4_6

[23] LIANG, Y.C., H.P. LEE, S.P. LIM, W.Z. LIN, K.H. LEE a C.G. WU. *Proper Orthogonal Decomposition and Its Applications—Part I: Theory.* Journal of Sound and Vibration [online]. 2002, 252(3), 527-544 [cit. 2023-03-24]. ISSN 0022460X. Dostupné z: doi:10.1006/jsvi.2001.4041

[24] LIN, Zhouchen, Minming CHEN a Yi MA. *The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices* [online]. 2010 [cit. 2023-05-03]. Dostupné z: doi:10.48550/arXiv.1009.5055

[25] MANN, Jordan a J. Nathan KUTZ. *Dynamic mode decomposition for financial trading strategies.* Quantitative Finance [online]. 2016, 16(11), 1643-1655 [cit. 2023-04-08]. ISSN 1469-7688. Dostupné z: doi:10.1080/14697688.2016.1170194

[26] MOHAMAD, Auday A.H. a Mohammed OSMAN. *Adaptive median filter background subtractions technique using fuzzy logic.* 2013 International Conference on Computing, Electrial and Electronic Engineering (ICCEEE) [online]. IEEE,

2013, 2013, 115-120 [cit. 2023-04-26]. ISBN 978-1-4673-6232-0. Dostupné z: doi:10.1109/ICCEEE.2013.6633917

[27] OLIVER, N.M., B. ROSARIO, A.P. PENTLAND a Chandrika KAMATH. *A Bayesian computer vision system for modeling human interactions.* IEEE Transactions on Pattern Analysis and Machine Intelligence [online]. 2004-1-7, 22(8), 831-843 [cit. 2023-04-28]. ISSN 01628828. Dostupné z: doi:10.1109/34.868684

[28] RAJMIC, Pavel. *Řídké a nízkohodnostní reprezentace signálů s aplikacemi.* Brno, 2014, 154 s. Habilitační práce. Vysoké učení technické v Brně.

[29] RODRIGUEZ, Paul a Brendt WOHLBERG. *Fast principal component pursuit via alternating minimization.* 2013 IEEE International Conference on Image Processing [online]. IEEE, 2013, 2013, 69-73 [cit. 2023-05-24]. ISBN 978-1-4799-2341-0. Dostupné z: doi:10.1109/ICIP.2013.6738015

[30] ROWLEY, Clarence W., Igor MEZIĆ, Shervin BAGHERI, Philipp SCHLATTER a Dan S. HENNINGSON. *Spectral analysis of nonlinear flows.* Journal of Fluid Mechanics [online]. 2009, 641, 115-127 [cit. 2023-03-24]. ISSN 0022-1120. Dostupné z: doi:10.1017/S0022112009992059

[31] SCHMID, Peter J. *Dynamic mode decomposition of numerical and experimental data.* Journal of Fluid Mechanics [online]. 2010, 656, 5-28 [cit. 2023-03-24]. ISSN 0022-1120. Dostupné z: doi:10.1017/S0022112010001217

[32] SCHMID, Peter J., Jonathan H. TU a Clarence W. ROWLEY. *Dynamic Mode Decomposition and Its Variants: Boundary Condition, Koopman, and Fourier Analyses.* Annual Review of Fluid Mechanics [online]. 2022, 54(1), 225-254 [cit. 2023-03-24]. ISSN 0066-4189. Dostupné z: doi:10.1146/annurev-fluid-030121-015835

[33] SHORES, Thomas S. *Applied linear algebra and matrix analysis* [online]. New York: Springer, 2007 [cit. 2023-05-14]. ISBN 978-0-387-33195-9. Dostupné z: https://doi.org/10.1007/978-0-387-48947-6

[34] SUNNY, K., A. SHEIKH a S. WAGH. *Application of Dynamic Mode Decomposition for Temperature Analysis in Smart Building.* 2020 7th International Conference on Control, Decision and Information Technologies (CoDIT) [online]. IEEE, 2020, 2020-6-29, 16(11), 1197-1202 [cit. 2023-04-08]. ISBN 978-1-7281-5953-9. ISSN 1469-7688. Dostupné z: doi:10.1109/CoDIT49905.2020.9263862

[35] TREFETHEN, Lloyd N. a David BAU. *Numerical linear algebra.* Philadelphia: Society for Industrial and Applied Mathematics, [1997]. ISBN 978-0-898713-61-9.

[36] YU, Yongchao a Jigen PENG. *The Moreau envelope based efficient first-order methods for sparse recovery.* Journal of Computational and Applied Mathematics [online]. 2017, 322(1), 109-128 [cit. 2023-05-05]. ISSN 03770427. Dostupné z: doi:10.1016/j.cam.2017.03.014

[37] YUAN, Xiaoming a Junfeng YANG. *Sparse and low rank matrix decomposition via alternating direction method* [online]. 2009 [cit. 2023-05-03]. Dostupné z: https://optimization-online.org/?p=10962

[38] ZÁVIŠKA Pavel, Ondřej MOKRÝ, Pavel RAJMIC. *S-SPADE Done Right: Detailed Study of the Sparse Audio Declipper Algorithms* [online].Brno University of Technology, Czech Republic, 2019.[cit. 2020-06-09]. Dostupné z: https://arxiv.org/pdf/1809.09847.pdf

[39] ZHANG, Dengsheng, Xing FU, Steve L. BRUNTON a N. Benjamin ERICHSON. *Wavelet Transform. Fundamentals of Image Data Mining* [online]. Cham: Springer International Publishing, 2019, 2019-05-14, 15(2), 35-44 [cit. 2023-04-18]. Texts in Computer Science. ISBN 978-3-030-17988-5. ISSN 1536-0040. Dostupné z: doi:10.1007/978-3-030-17989-2_3

[40] ZHANG, Fuzhen, Kit Ian KOU a Jifei MIAO. *Quaternions and matrices of quaternions.* Linear Algebra and its Applications [online]. 1997, 251, 21-57 [cit. 2023-05-11]. ISSN 00243795. Dostupné z: doi:10.1016/0024-3795(95)00543-9

[41] 8-bit Color Images. *Image processing learning resources* [online]. [cit. 2023-04-26]. Dostupné z: https://homepages.inf.ed.ac.uk/rbf/HIPR2/8bitcol.htm

[42] Closing. *Image Processing Learning Resources* [online]. 2003 [cit. 2023-05-25]. Dostupné z: https://homepages.inf.ed.ac.uk/rbf/HIPR2/close.htm

[43] Group (mathematics). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-05-10]. Dostupné z: https://en.wikipedia.org/wiki/Group_(mathematics)#CITEREFHerstein1975

[44] Pseudo-Inverse of a Matrix. In: *Hyper-Textbook: Optimization Models and Applications* [online]. EECS Department, UC Berkeley, 2021 [cit. 2023-05-14]. Dostupné z: https://inst.eecs.berkeley.edu/ ee127/sp21/livebook/def_pseudo_inv.html

[45] Structural similarity. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-06-21]. Dostupné z: https://en.wikipedia.org/wiki/Structural_similarity

[46] The Wiener filter. *Image Processing Learning Resources* [online]. 2003 [cit. 2023-05-25]. Dostupné z: https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/VELDHUIZEN/node15.html

[47] YCbCr. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-05-09]. Dostupné z: https://en.wikipedia.org/wiki/YCbCr#cite_ref-4

Seznam symbolů a zkratek

$\mathbb{N}$	množina přirozených čísel
$\mathbb{R}$	množina reálných čísel
$\mathbb{C}$	množina komplexních čísel
$\mathbb{H}$	množina kvaternionových čísel
$\mathbf{A}_{m,n}$	matice $\mathbf{A}$ řádu (m, n)
$\mathbf{A}^{+}$	Moore–Penroseova pseudoinverze matice $\mathbf{A}$
$\mathbf{A}^{*}$	hermitovská transpozice matice $\mathbf{A}$
$\overline{\mathbf{A}}$	matice komplexně sdružená k $\mathbf{A}$
$\mathbf{A}^{\top}$	transpozice matice $\mathbf{A}$
$\langle \mathbf{u}, \mathbf{v} \rangle$	skalární součin vektorů $\mathbf{u}, \mathbf{v}$
$\|\mathbf{u}\|$	norma vektoru $\mathbf{u}$
$\mathbf{M}$	označení matice vstupních dat
$\mathbf{L}$	nízkohodnostní matice statické složky
$\mathbf{S}$	řídká matice dynamické složky
SVD	singular value decomposition, singulární rozklad
DMD	dynamic mode decomposition, metoda dynamických modů
QDMD	kvatenionová metoda dynamických modů
mrDMD	multiresolution dynamic mode decomposition
PCA	principal component analysis; analýza hlavních komponent
RPCA	robust principal component analysis; robustní analýza hlavních komponent
PCP	principal component pursuit
PCP	kvaternionové principal component pursuit
prox_f	proximální operátor funkce f

px	pixel
soft	soft thresholding; měkké prahování
SVD	singular value decomposition; singulární rozklad
svt	singular value thresholding; prahování singulárních čísel
supp	nosič vektoru
vec	operátor vektorizace matice

A Seznam příloh

Z důvodu velikosti přiložených souborů je kompletní příloha dostupná pouze z Google disku VUT 200875:
`https://drive.google.com/file/d/1xQFBW0uX_V6QdMfnV-FqMMhX-voSC9Tw/view?usp=sharing`

A.1 Vstupní data

Ve složce `original_data` jsou přiložena reálná a simulovaná data, na kterých byly metody testovány. Je zde přiloženo:

`forest.jpg`	obrázek přírody použitý pro simulovaná data
`parot.png`	obrázek ptáčka použitý pro simulovaná data
`parot2.png`	obrázek druhého ptáčka použitý pro simulovaná data
`bees.mp4`	video včel
`street.mp4`	video ulice použité pro mrDMD

Výsledných 100 snímků simulovaných dat je přiloženo ve složce `imgs`. Snímky z prvního videa slunce jsou přiloženy ve složce `flare_prom` a snímky z druhého videa slunce jsou ve složce `flare_5x4`

A.2 Zdrojové kódy

Seznam zdrojových kódů

`color_dmd.m`	funkce obsahující QDMD metodu počítanou na kvaternionech
`color_dmd_complex.m`	funkce obsahující QDMD metodu počítanou na komplexních reprezentacích
`color_filtr_vykresleni.m`	funkce provádějící filtrované vykreslení výsledků pro barevná videa
`color_pcp.m`	funkce obsahující QPCP metodu počítanou na komplexních reprezentacích
`color_skalovane_vykresleni.m`	funkce provádějící škálované vykreslení pro barevná data
`color_vyhodnoceni.m`	funkce vyhodnocující přesnost metod pro barevná data
`color_vykresleni.m`	funkce provádějící vykreslení výsledků pro barevná data
`data_maker.m`	funkce vytvářející simulovaná data
`dmd.m`	funkce obsahující DMD metodu
`filtr_mrdmd_vykresleni.m`	funkce provádějící filtrované vykreslení výsledků pro metodu mrDMD
`filtr_vykresleni.m`	funkce provádějící filtrované vykreslení výsledků
`load_color_image.m`	funkce pro načtení barevné vstupní sekvence obrazů
`load_image.m`	funkce pro načtení obrazů ve stupních šedi
`load_video.m`	funkce pro načtení vstupního videa
`main.m`	skript obsahující volání jednotlivých funkcí
`median_filtr.m`	funkce obsahující mediánový filtr
`mrdmd.m`	funkce obsahující mrDMD metodu
`mrdmd_result.m`	funkce volající mrDMD a upravující výsledek
`nonconvex_pcp.m`	funkce obsahující nekonvexní variantu PCP
`nonconvex_pcp_char.m`	funkce obsahující RPCA dle Chartranda
`pcp.m`	funkce obsahující PCP metodu
`skalovane_mrdmd_vykresleni.m`	funkce provádějící škálované vykreslení pro metodu mrDMD
`skalovane_vykresleni.m`	funkce provádějící škálované vykreslení
`vyhodnoceni.m`	funkce vyhodnocující přesnost metod
`vykresleni.m`	funkce provádějící vykreslení výsledků

Všechny funkce vykreslení přehrají výsledek v MATLAB Movie Player a uloží jej do složky pod daným názvem ve formátu `avi`. Pro vyhodnocení pomocí funkce `vyhodnoceni.m` musíme znát obraz původního pozadí. Tato metoda vykreslí video SSIM mapy v MATLAB Movie Player a uloží ho do složky ve formátu `avi`.

A.3 Výsledky

Výsledky získané jednotlivými metodami jsou přiloženy ve složce `vysledky`. Ta je rozdělena na složky podle vstupních dat, které jsou rozděleny na složky podle jednotlivých metod (`noncnovex_pcp_char` je rozdělena na složky podle parametru p) a obsahují následující videa:

`color_filtr_vysledek.avi`	výsledek pro barevná videa je vykreslen pomocí Wienerova a morfologického filtru
`color_skalovany_vysledek.avi`	výsledek pro barevná videa je vykreslen pomocí škálování
`color_ssim.avi`	vykreslení SSIM mapy výsledku pro barevná videa
`color_vysledek.avi`	vykreslení neupraveného výsledku pro barevná videa
`filtr_vysledek.avi`	výsledek je vykreslen pomocí Wienerova a morfologického filtru
`skalovany_vysledek.avi`	výsledek je vykreslen pomocí škálování
`ssim.avi`	vykreslení SSIM mapy výsledku
`vysledek.avi`	vykreslení neupraveného výsledku