VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

# BAKALÁŘSKÁ PRÁCE

Brno, 2018 Vojtěch Juren


# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS


# OBNOVENÍ DVOU SIGNÁLŮ Z NEÚPLNÉHO POZOROVÁNÍ JEJICH KŘÍŽOVÉHO PROLNUTÍ
SIGNALS RECOVERY FROM TWO INCOMPLETE CROSSFADED SIGNALS

**BAKALÁŘSKÁ PRÁCE**
BACHELOR'S THESIS


**AUTOR PRÁCE** — **Vojtěch Juren**
AUTHOR

**VEDOUCÍ PRÁCE** — **Ing. Marie Mangová**
SUPERVISOR



**BRNO 2018**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Bakalářská práce

bakalářský studijní obor **Audio inženýrství**
Ústav telekomunikací

***Student:*** Vojtěch Juren ***ID:*** 174453
***Ročník:*** 3 ***Akademický rok:*** 2017/18

**NÁZEV TÉMATU:**

## Obnovení dvou signálů z neúplného pozorování jejich křížového prolnutí

**POKYNY PRO VYPRACOVÁNÍ:**

Nastudujte problematiku obnovení dvou signálů z neúplného pozorování jejich křížového prolnutí. Dále navrhněte vhodný optimalizační algoritmus pro řešení této úlohy v případě lineárního prolnutí dvou obrazových signálů, pro případ šumem a nelineárním prolnutím. Vše implementujete v prostředí Matlab, ověřte funkčnost implementace a proveďte analýzu, za jakých podmínek lze daný typ úlohy řešit, případně s jakou chybou.

**DOPORUČENÁ LITERATURA:**

[1] Žára, J.; Beneš, B.; Sochor, J.; Felkel, P.: Moderní počítačová grafika (2. vydání), Computer Press, 2005, ISBN 80-251-0454-0.

[2] Smith, S. W. : The scientist and engineer's guide to digital signal processing. San Diego, Calif.: California Technical Pub., c1997. ISBN 09-660-1763-3.

***Termín zadání:*** 5.2.2018 ***Termín odevzdání:*** 29.5.2018

***Vedoucí práce:*** Ing. Marie Mangová
***Konzultant:***

**prof. Ing. Jiří Mišurec, CSc.**
*předseda oborové rady*

## ABSTRAKT

Tato bakalářská práce se zabývá obnovením dvou obrazových signálů z jejich neúplného křížového prolnutí a obnovením stejných obrazových signálů z jejich neúplného křížového prolnutí obsahujícího šum. Vše je prakticky ověřeno a předvedeno pomocí softwaru Matlab.

## KLÍČOVÁ SLOVA

analýza hlavních komponent, konvexní kombinace, křížové prolnutí, lineární kombinace, šum

## ABSTRACT

This bachelor thesis deals with restoration of two signals from their cross-fade in limited section and restoration of two signals from their cross-fade in limited section containing noise. Everything is practically verified and performed with help of Matlab software.

## KEYWORDS

principal component analysis, convex combination, cross-fade, linear combination, noise

JUREN, Vojtěch. *Obnovení dvou signálů z neúplného pozorování jejich křížového prolnutí*. Brno, 2018, 41 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Marie Mangová

# PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Obnovení dvou signálů z neúplného pozorování jejich křížového prolnutí“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské 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 autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) 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 autora(-ky)

# PODĚKOVÁNÍ

Rád bych poděkoval vedoucí bakalářské práce paní Ing. Marii Mangové za odborné vedení, konzultace, mnoho trpělivosti a podnětné návrhy k práci.


Brno ..... .....

podpis autora(-ky)

# OBSAH

# ÚVOD

Při zpracování obrazů, videa nebo hudby a se zvyšující se digitalizací nachází čím dál větší uplatnění křížové prolnutí.

Obecně si křížové prolnutí můžeme představit jako přechodový jev mezi smísením libovolného počtu signálů. [1] Jinými slovy se dá říct, že jde o matematickou metodu násobení signálů čísly vždy z tolika vektorů, kolik je libovolných signálů. V mém případě se bude jednat o dva černobílé obrázky. V této práci chci z křížového prolnutí získat oba původní obrázky, a proto je potřeba použít optimalizačních algoritmů.

V kapitole 1 si uvedeme základy pro počítání celé problematiky. V kapitole 2 si ukážeme velikost a vlastnosti digitálních obrázků. Kapitola 3 uvádí, jak vytvořit lineární křížové prolnutí dvou černobílých obrázků pomocí konvexní kombinace a teoretické řešení obnovení dvou původních signálů. Kapitola 4 popisuje lineární křížové prolnutí dvou černobílých obrázků obsahující šum a teoretické řešení obnovení dvou původních signálů. V kapitole 5 si předvedeme praktické řešení obnovení dvou původních obrázků z jejich lineárního křížového prolnutí v prostředí Matlab a obnovení dvou obrázků z jejich křížového prolnutí obsahující šum také v prostředí Matlab. V kapitole 6 si uvedeme podmínky pro řešení a poslední kapitola 7 uvádí chyby řešení v obou případech obnovení obrázků.

# 1 VEKTOR A MATICE

Protože obraz počítač reprezentuje jako matici, nejdříve si definujeme, co matice, resp. vektor, je.

## 1.1 Vektor

Vektor definujeme takto [2]:

Mějme $k \in \mathbb{N}$. Uspořádanou $k$-tici reálných čísel $v_1, v_2, \ldots, v_k$

$$\vec{\mathbf{v}} = \big(v_1, v_2, \ldots, v_k\big)$$

nazýváme vektorem. Číslo $k$ značí rozměr vektoru $\vec{\mathbf{v}}$ a čísla $v_1, v_2, \ldots, v_k$ složky vektoru $\vec{\mathbf{v}}$.

## 1.2 Matice

Protože se velice často setkáme s případy, kdy bychom na zapsání vektoru potřebovali mnoho místa, raději tento zápis uvedeme do tvaru matice. Matici definujeme takto [2]:

Mějme reálnou matici typu $m \times n$

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}.$$

Předměty $a_{rs} \in \mathbb{R}\,(\,r = 1, 2, \ldots, m;\; s = 1, 2, \ldots, n;\,)$ nazýváme prvky (elementy) matice $\mathbf{A}$. První index $(r)$ je pořadovým číslem řádku, druhý index $(s)$ je pořadovým číslem sloupce, ve kterém je prvek $a_{rs}$. Je - li $m \neq n$, matice se nazývá obdélníková [také matice typu $(m;n)$]. Je - li $m = n$, matice se nazývá čtvercová [také matice řádu $n$ místo typu $(n;n]$. Matici typu $(n;1)$ nazýváme řádkovou maticí nebo řádkovým vektorem. Matici typu $(1;n)$ nazýváme sloupcovou maticí nebo sloupcovým vektorem. Matici typu (1;1), tj. řádu 1, nazýváme maticí bodovou.[2]

### 1.2.1 Sčítání a odčítání matic

Sčítání a odčítání matic stejných rozměrů definujeme po složkách, tedy pro $\mathbf{A}, \mathbf{B} \in Mat_{m\times n}(\mathbb{R})$ vyjde:

$$\mathbf{A} \pm \mathbf{B} = (a_{ij}) \pm (b_{ij}) = (a_{ij} \pm b_{ij}) \in Mat_{m\times n}(\mathbb{R}).$$

Ukázka výpočtu

$$\begin{pmatrix} 4 & 7 \\ 8 & 6 \\ 5 & 1 \end{pmatrix} + \begin{pmatrix} 1 & -3 \\ -2 & 4 \\ 3 & 8 \end{pmatrix} = \begin{pmatrix} 4+1 & 7-3 \\ 8-2 & 6+4 \\ 5-3 & 1+8 \end{pmatrix} = \begin{pmatrix} 5 & 4 \\ 6 & 10 \\ 2 & 9 \end{pmatrix}.$$

Příklad, kdy nelze sečítat nebo odečítat matice

$$\begin{pmatrix} 7 & 3 \\ 2 & 4 \end{pmatrix} \pm \begin{pmatrix} 8 & 1 & 6 \\ 9 & 6 & 5 \end{pmatrix}.$$

Nelze spočítat, protože matice mají různé rozměry.

### 1.2.2 Násobení matice číslem

Mějme matici typu $m \times n,$ kde $a_{rs} \in \mathbb{R}\,(\,r = 1, 2, \ldots, m;\, s = 1, 2, \ldots, n)$. Násobení matice $\mathbf{A}$ skalárem $\alpha \in \mathbb{R}$ definujeme tak, že každou složku matice vynásobíme skalárem $\alpha$, vznikne tedy:

$$\alpha\mathbf{A} = \begin{pmatrix} \alpha a_{11} & \alpha a_{12} & \ldots & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \ldots & \alpha a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha a_{m1} & \alpha a_{m2} & \ldots & \alpha a_{mn} \end{pmatrix}.$$

# 2 VELIKOST A VLASTNOSTI OBRÁZKŮ

Z 1 víme, že počítač reprezentuje obraz jako matici. Vlastnosti a velikosti obrázků jsou důležité pro výpočet výsledných hodnot matic, které budou určovat intenzitu černé a bílé barvy na jednotlivých pixelech obrázků. Určeme si jednotlivé vlastnosti černobílých obrázků, se kterými celou problematiku řeším.

## 2.1 Velikost obrázků

Matice obrázků mohou být jakkoliv rozměrné, ale nesmíme zapomínat, že z 1.2 plyne podmínka o stejných rozměrech matic, se kterými chceme počítat. Jestliže máme obrázek o rozměrech $m \times n$, pak $m$ určuje výšku (počet řádků) a $n$ šířku (počet sloupců) matice. Velikost matic obrázků tedy určují jejich rozlišení.

## 2.2 Vlastnosti obrázků

### 2.2.1 Hodnoty pixelů

Hodnoty všech pixelů (prvků matice obrázků) se odvíjí od datového typu, který je reprezentuje. Nejčastěji jsou binární a délky $k$, takže pixel může nabývat jakékoliv hodnoty $2^k$, kde $k$ závisí na bitové hloubce.[3]

### 2.2.2 Obrázky stupně šedi

Obrázky se ve stupni šedi skládají z jednoho kanálu, který reprezentuje intenzitu obrázku. Kladné hodnoty pixelů jsou žádoucí, protože představují intenzitu světelné energie, a proto nemohou být záporné. Nabývají hodnot $[0 \dots 2^k - 1]$. Například typický obrázek ve stupni šedi používá bitovou hloubku 8 bitů na pixel, což znamená intenzitu jednoho pixelu v rozsahu $0 - 255$. 0 představuje minimální intenzitu (černou) a 255 maximální intenzitu (bílou).[3]

## 2.3 Použité obrázky

Celá problematika obnovení dvou signálů z neúplného pozorování jejich křížového a obnovení dvou signálů z neúplného pozorování jejich křížového obsahujících šum je praktikována na následujících obrázcích.

Obr. 2.1: Ostružina.

Obr. 2.2: Pampeliška.

# 3 KŘÍŽOVÉ PROLNUTÍ

Jestliže chceme udělat prolnutí dvou obrázků, pak musíme každý obrázek vynásobit číslem v rozmezí $0 - 1$ a poté obrázky spolu sečíst. A to z toho důvodu, abychom viděli určitou část z jednoho obrázku a určitou část z druhého. V mém případě ale obrázky pozoruji v čase, jedná se tedy o křížové prolnutí, a proto obrázky musí být vynásobeny čísly z vektorů, přičemž každý vektor bude náležet jednomu obrázku. Dojde ke stavu, kdy na začátku vidíme jeden obrázek a na konci druhý. Protože jsou ale obrázky při každém násobení sečteny a vektory neobsahují čísla 0 a 1 ( v této problematice musí splňovat podmínky konvexní kombinace, což bude popsáno v 3.2.1), na začátku, resp. na konci násobení a sečítání bude vždy vidět malá část jednoho z obrázků. Nesmíme však udělat tu chybu, že bychom násobili vektor s obrázkem, protože postupně chceme vybírat jednotlivá čísla z vektorů a jimi násobit naše obrázky, aby ke křížovému prolnutí vůbec došlo.

Křížové prolnutí při konvexní kombinaci je speciálním případem lineární kombinace, proto si nejdříve ukážeme její definici v následující části.

## 3.1 Lineární kombinace vektorů

Lineární kombinaci definujeme takto [4]:

Mějme

$$\vec{\mathbf{v}}_1, \vec{\mathbf{v}}_2, \ldots, \vec{\mathbf{v}}_m \in \mathbb{R}^k \;\; a \;\; \alpha_1, \alpha_2, \ldots, \alpha_m \in \mathbb{R}.$$

Lineární kombinací rozumíme vektor:

$$\vec{\mathbf{w}} = \alpha_1\vec{\mathbf{v}}_1 + \alpha_2\vec{\mathbf{v}}_2 + \ldots + \alpha_m\vec{\mathbf{v}}_m = \sum_{k=1}^{m} \alpha_k\vec{\mathbf{v}}_k.$$

Například součet vektorů $2\vec{\boldsymbol{u}}$ a $3\vec{\boldsymbol{v}}$ se rovná

$$2\vec{\boldsymbol{u}} + 3\vec{\boldsymbol{v}} = 2\begin{pmatrix}4\\1\\2\end{pmatrix} + 3\begin{pmatrix}3\\1\\2\end{pmatrix} = \begin{pmatrix}2\cdot 4 + 3\cdot 3\\2\cdot 1 + 3\cdot 1\\2\cdot 2 + 3\cdot 2\end{pmatrix} = \begin{pmatrix}17\\5\\10\end{pmatrix} = \vec{\mathbf{w}}.$$

## 3.2 Křížové prolnutí při konvexní kombinaci

V mém případě křížové prolnutí při lineární kombinaci musí splňovat podmínky konvexní kombinace. To znamená, že pro vznik křížového prolnutí platí podmínka, kdy jeden koeficient pro násobení s jedním signálem roste od 0 do 1 a druhý koeficient pro

násobení s druhým signálem klesá od 1 do 0 a během každého násobení musí koeficienty dát dohromady součet 1. [5] Definici konvexní kombinace zapíšeme takto [6]:

Mějme

$$\vec{\mathbf{v}}_1, \vec{\mathbf{v}}_2, \ldots, \vec{\mathbf{v}}_m \in \mathbb{R}^k \;\; a \;\; \alpha_1, \alpha_2, \ldots, \alpha_m \in \mathbb{R}.$$

Konvexní kombinaci vektorů $\vec{\mathbf{v}}_1, \vec{\mathbf{v}}_2, \ldots, \vec{\mathbf{v}}_m$ nazveme lineární kombinací $\sum_{k=1}^{m} \alpha_k \vec{\mathbf{v}}_k$, která splňuje $\sum_{k=1}^{m} \alpha_k = 1, \alpha_k \geq 0, k \in \mathbb{N}$.

V úvodu této kapitoly 3 jsme si řekli, že obrázky násobíme čísly z vektorů. Pro matematickou představu křížového prolnutí si první obrázek označíme $\mathbf{X}$ a druhý $\mathbf{Y}$. Obecně tedy můžeme pro kombinace napsat tuto rovnici:

$$C_k = a_k \mathbf{X} + b_k \mathbf{Y}, \tag{3.1}$$

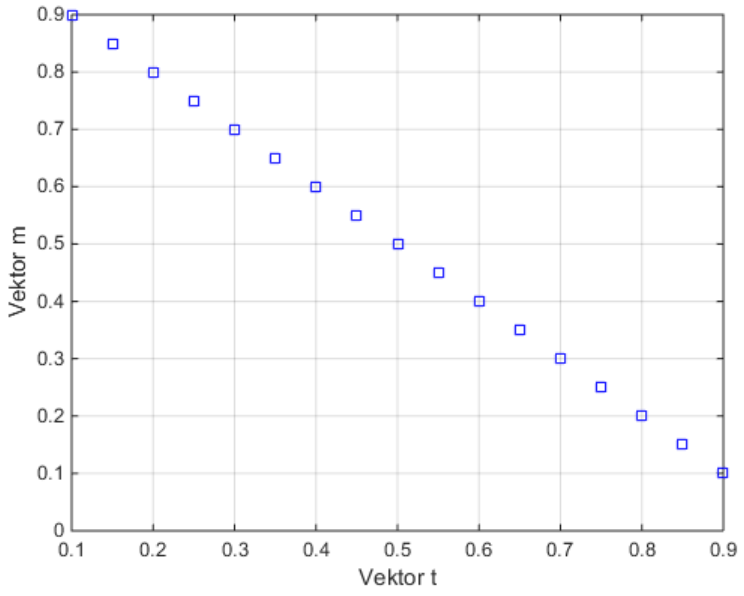
kde $C_k$ představuje výsledek $k$-té kombinace.

Protože koeficienty násobení dávají v součtu hodnotu 1 (jinak nelze splnit podmínky konvexní kombinace), můžeme rovnici upravit na tvar:

$$C_k = a_k \mathbf{X} + (1 - a_k)\mathbf{Y}. \tag{3.2}$$

### 3.2.1 Příklad prolnutí

Použité vektory v této problematice dle grafu jsou:




Obr. 3.1: Ukázka použitých vektorů.

Jak je vidět z hodnot vektorů, jejich složky nenabývají hodnot 0 a 1, přestože jsme si v předchozí definici uvedli podmínku $\alpha_k \geq 0$. To je proto, že pokud bychom již před prolnutím znali tyto koeficienty, nemuseli bychom obnovení obrázků řešit, protože bysme tím pádem znali původní obrázky.

Pro představu si ukážeme, jak matematicky vytvořit jedno prolnutí a jak následně toto prolnutí vypadá jako obrázek. Například vynásobíme číslo 0,5 z vektoru $\vec{\mathbf{t}}$ s maticí (obrázkem 5.3), kterou již máme označenou jako $\mathbf{X}$, a číslo 0,5 z vektoru $\vec{\mathbf{m}}$ vynásobíme s obrázkem 5.5 označeným jako $\mathbf{Y}$. Pro index $k$ platí $k = 9$ a určuje pořadí prvků ve vektorech $\vec{\mathbf{t}}$ a $\vec{\mathbf{m}}$. Matice obrázků jsou typu $m \times n$, kde $m = 3024$ a $n = 4032$. Vznikne tato rovnice:

$$C_9 = t_9 \cdot \mathbf{X} + (1 - t_9) \cdot \mathbf{Y}$$

Dosadíme obecné prvky do matic, čímž vznikne tato rovnice:

$$C_9 = 0{,}5 \cdot \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix} + 0{,}5 \cdot \begin{pmatrix} y_{11} & y_{12} & \dots & y_{1n} \\ y_{21} & y_{22} & \dots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m} & y_{m2} & \dots & y_{mn} \end{pmatrix}$$

Po dosazení konkrétních čísel z matic obrázků dostaneme:

$$C_9 = 0{,}5 \cdot \begin{pmatrix} 114 & 114 & \dots & 63 \\ 115 & 116 & \dots & 63 \\ \vdots & \vdots & \ddots & \vdots \\ 8 & 9 & \dots & 133 \end{pmatrix} + 0{,}5 \cdot \begin{pmatrix} 134 & 134 & \dots & 58 \\ 125 & 128 & \dots & 52 \\ \vdots & \vdots & \ddots & \vdots \\ 198 & 197 & \dots & 169 \end{pmatrix}$$

Sečtením těchto dvou matic vznikne obrázek (kombinace) 3.2:

V mém případě mám několik takových prolnutí k dipozici. Při snaze obnovení prvního z obrázků jsem pak odečetl kombinaci číslo jedna od kombinace číslo dvě a tento rozdíl postupně přičítal ke kombinaci číslo dvě a podle stanovených podmínek, kdy minimální hodnota musí být větší nebo rovna 0 a maximální hodnota menší nebo rovna 255, se určuje, jestli je dosaženo obnoveného obrázku. Při snaze obnovení druhého z obrázků jsem znovu odečetl kombinaci číslo jedna od kombinace číslo dvě a tento rozdíl postupně odečítal od druhé kombinace a tak se obnovil druhý obrázek. Více je tento postup popsán v kapitole 5 .

Obr. 3.2: Výsledek součtu součinů čísla 0,5 z vektoru $\vec{\mathbf{t}}$ s obrázkem $\mathbf{X}$ a čísla 0,5 z vektoru $\vec{\mathbf{m}}$ s obrázkem $\mathbf{Y}$.

# 4 KŘÍŽOVÉ PROLNUTÍ OBSAHUJÍCÍ ŠUM

Obsahem zadání práce bylo také vytvořit křížové prolnutí obsahující šum a získat zpět původní obrázky. Jedná se o případ, kdy kombinace obrázků jsou zašuměné. Pro odstranění šumu jsem použil optimalizačního algoritmu ve formě analýzy hlavních komponent, kterou se zabývá následující podkapitola. Porovnejme obrázek 3.2 jako výsledek součtu součinů čísla 0,5 z vektoru $\vec{\mathbf{t}}$ s obrázkem $\mathbf{X}$ a čísla 0,5 z vektoru $\vec{\mathbf{m}}$ s obrázkem $\mathbf{Y}$ se stejným obrázkem obsahující navíc šum 4.1:

Obr. 4.1: Výsledek součtu součinů čísla 0,5 z vektoru $\vec{\mathbf{t}}$ s obrázkem $\mathbf{X}$ a čísla 0,5 z vektoru $\vec{\mathbf{m}}$ s obrázkem $\mathbf{Y}$. Jejich součet navíc obsahuje šum.

Pro výpočet konvexní kombinace s nějakou chybou, kterou je šum, platí rovnice:

$$C_k = t_k\mathbf{X} + (1 - t_k)\mathbf{Y} + \mathbf{n}_k,$$

kde $k$ představuje pozici daného čísla ve vektoru $\vec{\mathbf{t}}$ a $\mathbf{n}$ je matice šumu o velikosti matic $\mathbf{X}$ a $\mathbf{Y}$. Dosazením prvků z vektoru $\vec{\mathbf{t}}$ dostaneme tuto soustavu rovnic:

$$C_1 = t_1\mathbf{X} + (1 - t_1)\mathbf{Y} + \mathbf{n}_1$$

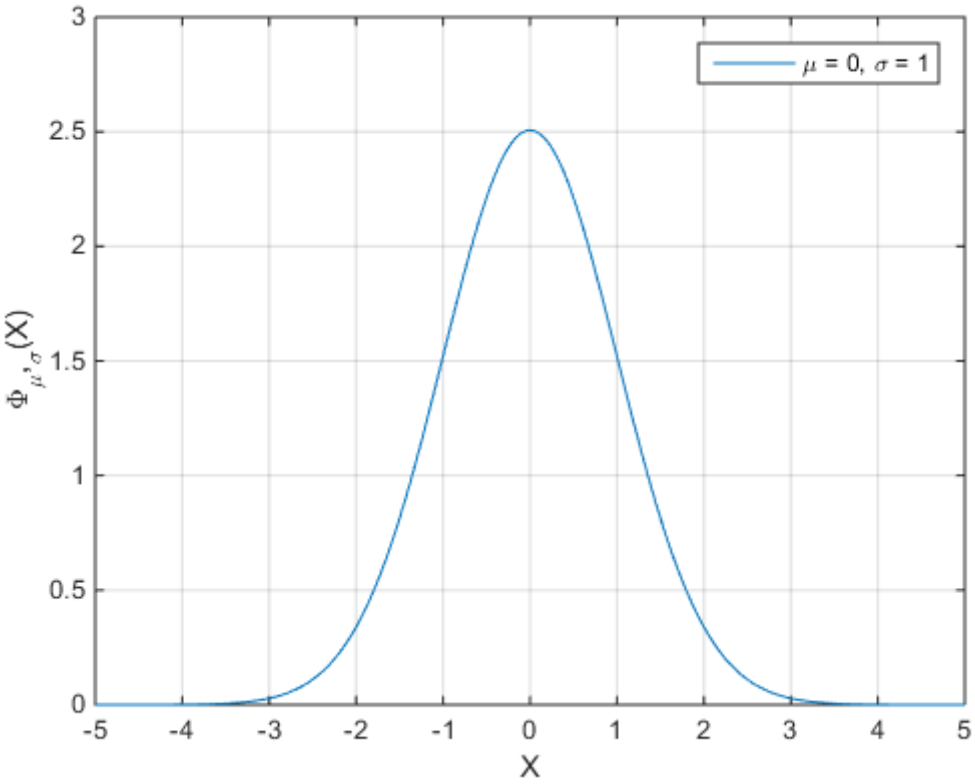
$$C_2 = t_2\mathbf{X} + (1 - t_2)\mathbf{Y} + \mathbf{n}_2$$

$$\vdots$$

$$C_k = t_k\mathbf{X} + (1 - t_k)\mathbf{Y} + \mathbf{n}_k$$

V mém případě je šum s Gaussovým rozdělením, známým také jako normálním. Střední hodnota je $\mu = 0$ a směrodatná odchylka $\sigma = 1$. [7] Funkce popisující tento typ rozdělení a obrázek toho rozdělení jsou následující [7]:

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{4.1}$$




Obr. 4.2: Obrázek normálního rozdělení.

## 4.1 Analýza hlavních komponent

Abychom téměř nebo alespoň částečně odstranili šum z jednotlivých kombinací, musíme provést analýzu hlavních komponent, kterou v anglické literatuře najdeme pod zkratkou PCA (Principal Component Analysis). Tato analýza pomáhá zobrazit mnohorozměrná data na takovou úroveň, aby se data dala reprezentovat do trojrozměrného, případně méně rozměrného zobrazení bez ztráty většiny informací. Pokud totiž máme zkoumanou matici o rozměrech $n \times m$, platí, že každé $m$ tvoří jednu souřadnicovou osu. Proto se při $m$ větším než 3 tato data nedají normálně zobrazit. Musí tedy dojít k redukci dimenzí, která se také hodí právě pro případy se zašuměnými daty. [8]

Techniku redukce dimenzí lze popsat jako metodu lineární transformace původních znaků na nové, nekorelované proměnné, nazvané hlavní komponenty. Každá hlavní komponenta představuje lineární kombinaci původních znaků. Základní charakteristikou každé hlavní komponenty je její míra variability čili rozptyl. Hlavní komponenty jsou seřazeny dle důležitosti, tj. dle klesajícího rozptylu od největšího k nejmenšímu. Většina informace o variabilitě původních dat je přitom soustředěna do první komponenty a nejméně informace je obsaženo v poslední komponentě. Platí pravidlo, že má-li nějaký původní znak malý, nebo dokonce nulový rozptyl, není schopen přispívat k rozlišení mezi objekty.[9]

Pokud každý sloupec dat zašuměné matice obsahuje nezávisle rozložený Gaussovský šum, pak nově vzniklá matice vytvořená součinem zašuměné matice a z ní vypočítané matice přes PCA bude také obsahovat stejně rozložený Gaussovský šum. Prvních několik komponent obsahuje nejvíce rozptylu z celkového rozptylu dat, čímž dosahují největšího poměru signálu k šumu (SNR).[10] Konkrétně první komponenta má největší SNR, druhá méně než první, třetí méně než druhá a tak dále.

PCA je matematický postup, který se skládá buď z výpočtu kovarianční matice a jejích vlastních čísel a vlastních vektorů, nebo ze singulárního rozkladu zkoumané matice. Software Matlab, ve které jsem problematiku řešil, ve výchozím nastavení při použití PCA nejprve centruje vstupní data a poté vypočítá jejich singulární rozklad. [11]

V následujících podkapitolách si ukážeme, jak PCA matematicky funguje.

### 4.1.1 Centrování dat

Prvním krokem při výpočtu PCA je centrování dat. V mém případě mám 17 kombinací. Vytvořil jsem novou matici, kde každý řádek je zvektorizovaná kombinace a od ní odečtený aritmetický průměr ze všech prvků příslušné kombinace. Matematicky zapíšeme takto:

$$\vec{A}_{ij} - \frac{1}{n}\sum_{j=1}^{n} a_{ij},$$

kde $i$ představuje řádek sloupcového vektoru z $\vec{A}_{ij}$, $j$ je pořadí sloupce a $n$ celkový počet sloupců. Dále jsem v Matlabu dal příkaz, aby každý takový nově vycentrovaný vektor dal do řádku. Vznikla tedy matice $n \times m$, kde $n = 17$, což je počet kombinací, a $m$ je celkový počet prvků v každém z vektorů jednotlivé kombinace.

### 4.1.2 Singulární rozklad

Z 4.1 víme, že Matlab ve výchozím nastavení počítá analýzu hlavních komponent přes singulární rozklad. Proto si tento postup nyní popíšeme podle[12]:

Singulární rozklad (singular value decomposition, SVD) je definován pro libovolnou obdélníkovou matici $\mathbf{A} \in \mathbb{R}^{n\times p}$ . Je-li $r \leq \min\{\text{n,p}\}$ hodnost matice $\mathbf{A}$, pak má tvar:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T, \tag{4.2}$$

kde $\mathbf{U} \in \mathbb{R}^{n\times n}$, $\mathbf{V} \in \mathbb{R}^{p\times p}$ jsou ortonormální matice, tj.$\mathbf{U}^T\mathbf{U} = \mathbf{I}_n$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}_p$ a matice $\Sigma \in \mathbb{R}^{n\times p}$ má na prvních $r$ diagonálních pozicích prvky

$$\sigma_{ii} > 0, \quad i = 1, \ldots, r \tag{4.3}$$

a je nulová jinde. Čísla $\sigma_{ii}$ se nazývají singulární čísla a platí, že se rovnají odmocnině nenulových vlastních čísel matice $\mathbf{A}\mathbf{A}^T$ (i $\mathbf{A}^T\mathbf{A}$ ). Jsou tedy vždy reálná a kladná. Je konvencí, aby singulární čísla byla na diagonále $\Sigma$ uspořádána sestupně. Sloupce matice $\mathbf{U}$ obsahují takzvané levé singulární vektory a jsou zároveň vlastními vektory matice $\mathbf{A}\mathbf{A}^T$. Sloupce $\mathbf{V}$ se nazývají pravé singulární vektory a jsou i vlastními vektory $\mathbf{A}^T\mathbf{A}$. Není těžké vidět, že pro symetrické matice ($\mathbf{A} = \mathbf{A}^T$) představuje spektrální a singulární rozklad totéž. V lineární algebře se pro symetrické matice používá vždy pojem spektrální rozklad, kdežto statistická literatura používá i pojem singulární rozklad. Ze singulárního rozkladu 4.2 dostaneme po jednoduchém (ale dlouhém) rozepsání po prvcích ekvivalentní vyjádření:

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_{ii}\vec{\mathbf{u}}_i\vec{\mathbf{v}}_i^T. \tag{4.4}$$

Pro jednotlivé členy přitom platí $||\sigma_{ii}\vec{\mathbf{u}}_i\vec{\mathbf{v}}_i^T|| = \sigma_{ii}$ a tudíž

$$||\sigma_{11}\vec{\mathbf{u}}_1\vec{\mathbf{v}}_1^T|| = \sigma_{11} \geq ||\sigma_{22}\vec{\mathbf{u}}_2\vec{\mathbf{v}}_2^T|| = \sigma_{22} \geq \cdots \geq ||\sigma_{rr}\vec{\mathbf{u}}_r\vec{\mathbf{v}}_r^T|| = \sigma_{rr}. \tag{4.5}$$

Ve vztahu 4.4 nahlížíme na matici $\mathbf{A}$ jako na součet celkového počtu $r$ komponent. Jde vlastně o ekvivalentní vyjádření pozorovaných dat vůči jiným ortonormálním bázím. Lze říci, že komponenty příslušné největším singulárním číslům mají v pozorovaných datech největší váhu.

Statistické metody pro redukci dimenze založené na singulárním (popř. pro symetrické pozitivně semidefinitní matice spektrálním) rozkladu určité matice $\mathbf{A}$ lze interpretovat i tak, že samotnou matici $\mathbf{A}$ nahradíme aproximací podle:

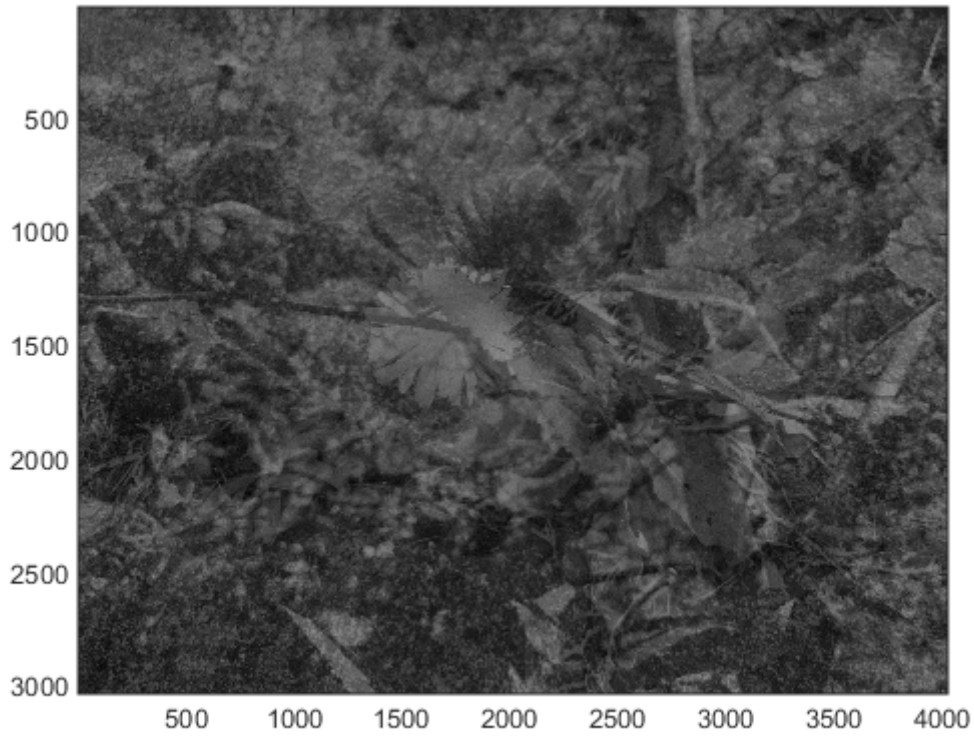
$$\mathbf{A} \approx \sum_{i=1}^{s} \sigma_{ii}\vec{\mathbf{u}}_i\vec{\mathbf{v}}_i^T, \tag{4.6}$$

v němž $s < r$. To znamená, že se zcela ignoruje vliv takových komponent z 4.4, které přísluší nejmenším (a tedy v určitém smyslu nejméně důležitým) singulárním číslům.

### 4.1.3 Shrnutí řešení zbavení šumu

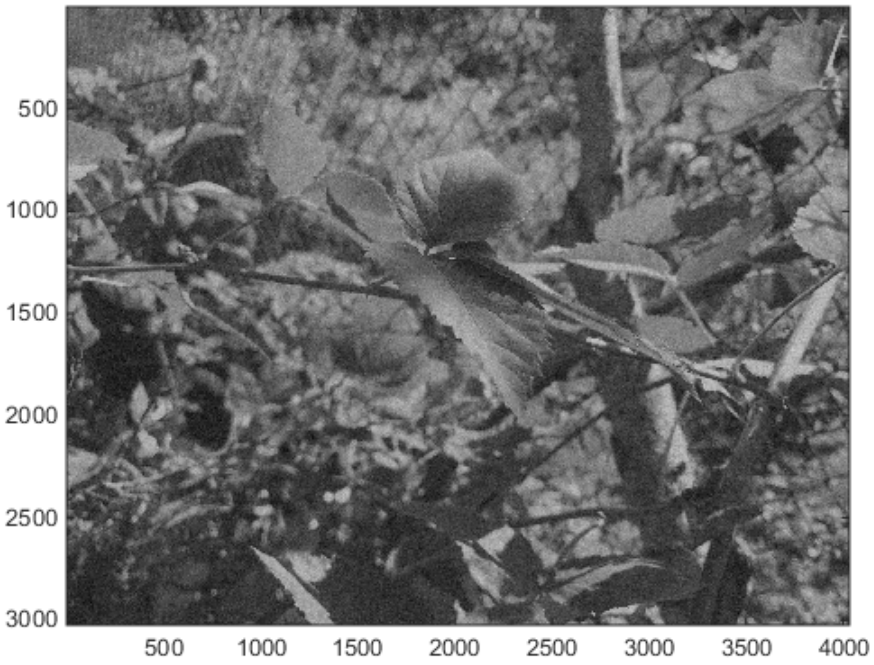
Singulárním rozkladem tedy získáme hlavní komponenty. Myšlenka odšumění je taková, že se vezme první komponenta (vektor), která obsahuje největší poměr signálu od šumu, a vynásobí se se zašuměným obrázkem (maticí). Pro představu matematicky zapíšeme takto: $\mathbf{T}_r = \mathbf{A}\mathbf{V}_r$, kde $\mathbf{T}_r$ je vzniklá matice, $\mathbf{A}$ je zašuměná matice a $\mathbf{V}_r$ je matice hlavních komponent. V případě vynásobení první komponentou pro $r$ platí $r = 1$, takže z matice $\mathbf{T}_r$ se stane vektor $\vec{T}_r$. Protože vynásobení zašuměné matice (obsahující všechny kombinace) s první komponentou je projekce zašuměné matice na tuto komponentu a chceme opět zpátky získat všechny kombinace, musíme nově vzniklý vektor $\vec{T}_r$ transformovat do původního souřadného systému. V mém případě ale při použití pouze první komponenty vzniklo několik podobných obrázků, které se nedají využít pro navržený algoritmus obnovování. Takto například vypadaly první a poslední kombinace 4.3 , 4.4 :


Obr. 4.3: První kombinace při použití první komponenty.

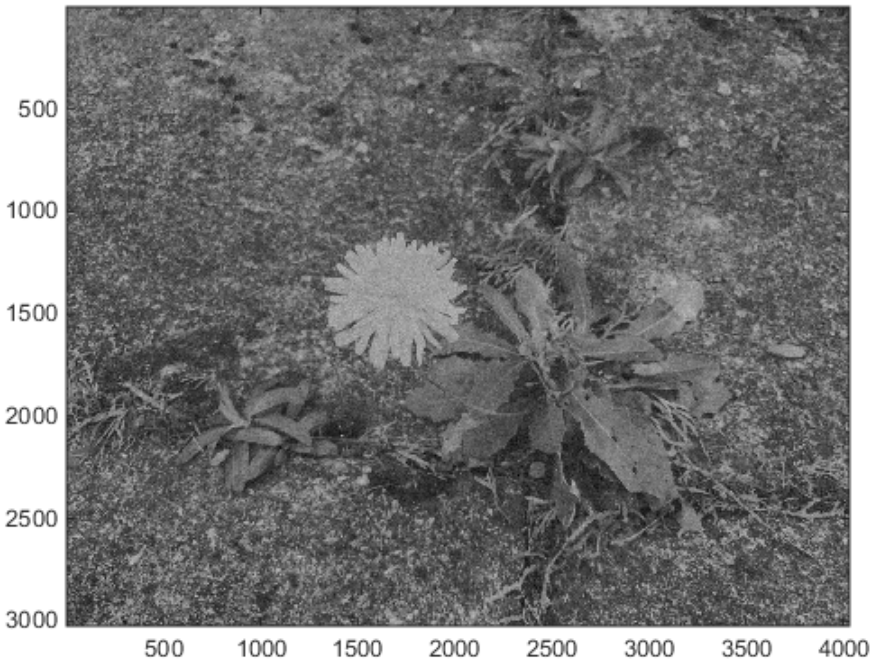



Obr. 4.4: Sedmnáctá kombinace při použití první komponenty.

Je tedy zřetelné, že obrázky jsou v podstatě totožné, akorát mají rozdílnou světlost. Proto jsem vzal první dvě komponenty, na ně promítl zašuměnou matici a vrátil zpět do jejich souřadného systému. První a poslední kombinace pak vypadají takto 4.5 , 4.6 :




Obr. 4.5: První kombinace při použití prvních dvou komponent.




Obr. 4.6: Sedmnáctá kombinace při použití prvních dvou komponent.

Zároveň si můžeme všimnout, že tyto obrázky jsou výrazně zbaveny šumu oproti zašuměné kombinaci 4.1 . Pro obnovení původních obrázků jsem opět použil stejný algoritmus jako při obnovování nezašuměných kombinací, ale bylo potřeba provést drobné změny. Tento postup je více popsán v následující kapitole.

# 5 ŘEŠENÍ PROBLEMATIKY V PROSTŘEDÍ MATLAB

V poslední kapitole si uvedeme algoritmus pro vytvoření křížového prolnutí dvou obrázků a především jejich obnovení do původních podob.
Křížové prolnutí znamená, že na začátku prolínání se uživateli zobrazí jeden obrázek a postupně se mu bude vykreslovat kombinace jednoho a druhého obrázku, až se na konci prolínání zobrazí obrázek druhý. S jistotou můžeme říci, že křížové prolnutí proběhlo minimálně dvakrát. Pokud by proběhlo jednou, jednalo by se pouze o prolnutí. Musíme tedy rozlišovat prolnutí a křížové prolnutí. Z prvních dvou prolnutí (kombinací) tedy můžeme vytvořit výchozí matici, kterou vynásobíme určitým koeficientem a tento součin přičteme k matici prolnutí (kombinace) číslo dva. Pokud nesplníme podmínky, že maximální hodnota kteréhokoliv prvku matice vzniklé součtem součinu matice číslo dva a výchozí matice násobené koeficientem nebude větší než 255 nebo minimální hodnota kteréhokoliv prvku matice nebude nižší než 0, potom koeficient pro násobení zvýšíme o libovolnou hodnotu. Tu ale raději zvolíme nízkou např. 0,1, abychom zajistili co nejvyšší přesnost výpočtu maximálních a minimálních hodnot. Celý proces opakujeme znovu, dokud alespoň jedna z podmínek nebude platná. Jakmile alespoň jedna z podmínek platí, znamená to, že jsme dosáhli maximální hodnoty vyšší než 255, resp. minimální hodnoty nižší než 0. Tyto hodnoty určují meze, kterých mohou nabývat pixely v obrázcích s bitovou hloubkou 8 bitů. Poté vytvoříme takový algoritmus, kdy vezmeme předchozí hodnotu koeficientu násobení s výchozí maticí a tedy bude platit, že maximální a minimální hodnoty prvků matice obnovovaného obrázku jsou v mezích $0 - 255$ včetně. Takovým postupem získáme oba obrázky.

### 5.0.4 Vytvoření křížového prolnutí

Pro vytvoření křížového prolnutí načteme obrázky, které převedeme do stupně šedi, jelikož je tato problematika s takovými obrázky řešena. Následně tyto šedé obrázky převedeme do datového typu double, aby se hodnoty prvků matic obou obrázků přepsaly do binárního kódu, a díky tomu bude celkový výpočet probíhat rychleji. Poté definujeme vektor, který je lineárně rostoucí s krokem 0,05 a délkou sedmnácti prvků. V mém případě nabývá hodnot od 0,1 do 0,9. Z hodnot je patrné, že jsou větší než 0 a menší než 1. To proto, abychom splnili podmínky konvexní kombinace z podkapitoly 3.2. Dále definujeme druhý vektor, jehož hodnoty jsou lineárně klesající s krokem 0,05 oproti prvnímu vektoru. První vektor si označíme t a druhý m. Zápis kódu v softwaru Matlab vypadá takto:

Výpis 5.1: `nacteni_obrazku_a_prealokace.m`

```
obr1=imread('ostruzina.jpg');
obr2=imread('pampeliska.jpg');
sedaobr1=rgb2gray(obr1);
sedaobr2=rgb2gray(obr2);
doubleobr1=double(sedaobr1);
doubleobr2=double(sedaobr2);

t=0.1:0.05:0.9;
m=1-t;
C=zeros(size(doubleobr1));
```

Pro vznik jednotlivých prolnutí použijeme cyklus for, kterému určíme, kolikrát se má vypočítat. Počet cyklů určíme délkou jednoho z vektorů. Dále vytvoříme příkaz, aby do první matice prolnutí zapsal do všech řádků a všech sloupců výsledek součtu součinů prvního prvku vektoru t s maticí typu double prvního obrázku a prvního prvku vektoru m s maticí typu double druhého obrázku. Po tomto výpočtu se uživateli zobrazí výsledek ve formě obrázku. Ještě nastavíme dobu, na jakou se obrázek uživateli zobrazí a cyklus ukončíme.

Výpis 5.2: `cyklus_pro_tvorbu_kombinaci.m`

```
for w = 1:length(t);
C(:,:,w)=t(w)*doubleobr1 + m(w)*doubleobr2;
imshow(C(:,:,w),[]);
pause(0.2)
end
```

Protože se cyklus vykoná tolikrát, jaká je velikost jednoho z vektorů a při jednotlivých cyklech se vytváří různé kombinace obrázků, z matice, kam se kombinace zapisují, vznikne 3D matice.
Příklady různých prolnutí jsou uvedeny na obrázcích 3.2, 5.1 a 5.2:

Obr. 5.1: Výsledek součtu součinů čísla 0,3 z vektoru $\vec{\mathbf{t}}$ s obrázkem $\mathbf{X}$ a čísla 0,7 z vektoru $\vec{\mathbf{m}}$ s obrázkem $\mathbf{Y}$.

Obr. 5.2: Výsledek součtu součinů čísla 0,7 z vektoru $\vec{\mathbf{t}}$ s obrázkem $\mathbf{X}$ a čísla 0,3 z vektoru $\vec{\mathbf{m}}$ s obrázkem $\mathbf{Y}$.

### 5.0.5 Vytvoření křížového prolnutí se šumem a PCA

Pro křížové prolnutí se šumem platí stejný postup naprogramování s tím rozdílem, že šum musíme dodat. To, jak moc se šum v kombinacích projeví, určuje hodnota směrodatné odchylky. Já jsem zvolil hodnotu 30. Ke každé nově vzniklé kombinaci

tedy vždy přičteme náhodná čísla s normálním rozdělením vynásobená směrodatnou odchylkou. Kód potom vypadá takto:

Výpis 5.3: `cyklus_pro_tvorbu_kombinaci_sum.m`

```
noisy=zeros(size(doublekytka));
so = 30;

for e = 1:length(t)
noisy(:,:,e) = (t(e)*(doublelod)+ m(e)*(doublekytka))+(randn(size(doublekytka))*so
imshow(noisy(:,:,e),[]);
pause(0.5)
end
```

Celý další postup je naprogramován v přiloženém skriptu `odsumeni_a_zobrazeni.m` Protože máme 3D matici obsahující 17 kombinací, pro analýzu hlavních komponent tyto kombinace vložíme do 2D matice. Pomocí cyklu for tedy nejdřív všechny kombinace (3D matici) označíme jako N. Poté každou jednotlivou kombinaci zvektorizujeme a odečteme od ní průměr všech jejích prvků. Tím dojde k vycentrování a máme tedy 2D matici o 17 řádcích a počtu prvků zvektorizovaných kombinací. Následně provedeme PCA z transponované vycentrované matice. Ve spoustě příkladů jsem totiž viděl matici před analýzou transponovat. Určíme si proměnné, které chceme z PCA získat. Já jsem je pojmenoval `coeff` a `score`. Proměnná `coeff` obsahuje seřazené hlavní komponenty od největší po nejmenší. Proměnná `score` představuje zašuměnou matici N v prostoru hlavních komponent. [11] Protože jsme si řekli, že prvních několik komponent obsahuje největší poměr signálu od šumu, matici `score` vynásobíme s proměnnou `coeff` obsahující první dvě komponenty definované jako m = 1 a n = 2, čímž se zbavíme většiny šumu. Také dojde ke projekci na tyto dvě komponenty. Tuto projekci si označíme jako proměnnou `proj`. Když jsem tuto projekci použil při obnovovacím algoritmu, software Matlab ukazoval chybu v podmínce prvního cyklu while. Bylo tedy jasné, že hodnoty vstupní matice do cyklu byly mimo rozsah. Proto jsem každou kombinaci přeškáloval do těchto mezí. Nakonec zbývá každý sloupec obsahující jednu z kombinací převést zpět do souřadného systému a zobrazit si, jak vypadají. Nyní jsou připraveny pro obnovovací algoritmus.

## 5.1 Obnovení obrázků bez šumu

Celý následující popis je naprogramován v přiloženém skriptu `obnoveni_cista.m` . Algoritmus na obnovení obrázků je navržený tak, že k matici kombinace (prolnutí) číslo dva bude přičítat výchozí matici, resp. od kombinace číslo dva bude odečítat výchozí matici. Výchozí matici vypočítáme jako rozdíl prvních dvou kombinací. S jistotou můžeme říci, že křížové prolnutí proběhlo minimálně dvakrát, protože mám

17 kombinací. Pokud by proběhlo jednou, jednalo by se pouze o prolnutí. Dále si definujeme koeficient pro násobení s výchozí maticí a tento součin budeme v rámci cyklu while sečítat s maticí druhé kombinace. Cyklus while si popíšeme v dalším odstavci. Přičítání, resp. odečítání skončí, až přestanou platit obě podmínky, kterými jsou maximální a minimální hodnoty prvků matice, která bude výsledkem součtu matice druhé kombinace a výchozí matice násobené koeficientem. Dále prealokujeme matici, která bude výsledkem součtu matice druhé kombinace a výchozí matice násobené koeficientem, aby program počítal rychleji. Abychom mohli uživatele informovat, kolikrát přičtení proběhlo, před samotným cyklem while vytvoříme proměnnou s hodnotou nula, protože v rámci přičítání k této proměnné přičteme hodnotu 1 a uživateli se zobrazí počet cyklů, které proběhly.
Protože předem nevíme, kolikrát se matice budou sečítat, zvolíme cyklus while, který počítá do doby, než přestane platit jeho podmínka. Také ale nevíme, jestli se dříve dostaneme k hodnotě 0 nebo 255 určujících meze, kterých mohou nabývat pixely v obrázcích s bitovou hloubkou 8 bitů podle 2. Proto cyklus bude počítat s oběma podmínkami maximální a minimální hodnoty matice vzniklé součtem. V rámci cyklu tedy napíšeme příkaz pro součet matice druhé kombinace s výchozí maticí násobené koeficientem. Určíme maximální a minimální hodnoty tohoto součtu a pro případ, že by obě podmínky platily, ke koeficientu násobení přičteme hodnotu 0,05, protože jsme zatím nedosáhli původního obrázku. Hodnota ale může být jiná. Čím nižší je, tím se získá vyšší přesnost rekonstrukce obrazu. K proměnné pro informování uživatele o počtu cyklů přičteme hodnotu 1, aby věděl, po kolikáté cyklus proběhl. Jakmile obě podmínky přestanou platit, znamená to, že jsme přesáhli hodnoty 0 a 255. Následně abychom měli originální obrázek, vezmeme předchozí hodnotu koeficientu a, který vynásobíme s výchozí maticí, součin přičteme k matici druhé kombinace, a tak získáme původní obrázek, jelikož budeme v mezích $0 - 255$. Uživatele informujeme o dosažení obrázku, výslednou matici převedeme do typu uint8 pro přepis hodnot z typu double na celočíselné hodnoty.[13] Uživateli zobrazíme dosažený obrázek a pozastavíme jej na zvolenou dobu. Stejný postup provedeme pro druhý obrázek s tím rozdílem, že od matice druhé kombinace budeme výchozí matici odečítat. Při dosažení druhého obrázku nakonec uživateli zobrazíme oba obnovené obrázky vedle sebe.
Pro porovnání si ukažme originální 5.3 , 5.5 a obnovené obrázky 5.4 , 5.6 :

Obr. 5.3: Originální obrázek ostružiny.

Obr. 5.4: Obnovený obrázek ostružiny.

Obr. 5.5: Originální obrázek pampelišky.

Obr. 5.6: Obnovený obrázek pampelišky.

### 5.1.1 Obnovení obrázků obshaující šum

Při obnovování obrázků se nabízí použití stejného algoritmu pro obnovování obrázků bez šumu. Ale jak je zmíněno v 5.0.5, tento pokus nebyl úspěšný. Proto do-

šlo k přeškálování hodnot. Pak už jen stačí použít algoritmu pro obnovování a při každém zpětném kroku v cyklech while do mezí $0-255$ zaměnit logický operátor || (nebo) za logický operátor  (a zároveň). Nedokážu vysvětlit, z jakého důvodu tomu tak je, ale tento způsob funguje. Když se uživateli zobrazí zpráva o dosažení každého z obrázků, je třeba před převodem na celočíselné hodnoty pomocí příkazu `uint8` přeškálovat obrázky zpět. Nakonec se uživateli zobrazí oba částečně odšuměné obrázky a program je u konce.
Pro porovnání si ukažme originální 5.3 , 5.5  a obnovené obrázky 5.7 , 5.8 :

Obr. 5.7: Obnovený obrázek ostružiny ze zašuměných kombinací.

Obr. 5.8: Obnovený obrázek pampelišky ze zašuměných kombinací.

# 6 PODMÍNKY ŘEŠENÍ

Pro podmínky řešení problematiky křížového prolnutí bez šumu jsem zjistil, že obrázky (matice) pro tvorbu kombinací musí mít stejné rozměry a musí splňovat podmínky konvexní kombinace. To stejné platí pro křížové prolnutí se šumem, ale navíc pro odšumění je potřeba použít minimálně prvních dvou hlavních komponent.

# 7 CHYBY ŘEŠENÍ

Pro případ obnovování nezašuměných kombinací jsem vypočítal tyto relativní chyby: Ostružina - chyba mezi maximální hodnotou originálního obrázku a obnoveného je 0%, chyba mezi minimální hodnotou originálního obrázku a obnoveného je 15,25%. Pampeliška - chyba mezi maximální hodnotou originálního obrázku a obnoveného je 0%, chyba mezi minimální hodnotou originálního obrázku a obnoveného je 13 %. Způsob výpočtu je následující: $\delta_{ost} = \frac{M_{max} - X_{max}}{X_{max}} 100$, kde $\delta_{ost}$ je výsledná relativní chyba pro obnovený obrázek ostružiny, $M_{max}$ je maximální hodnota matice obnoveného obrázku ostružiny a $X_{max}$ je maximální hodnota původního obrázku ostružiny. Tímto způsobem jsem spočítal relativní chyby pro maximální hodnoty pampelišky a minimální hodnoty pro ostružiny i pampelišky.

# 8 ZÁVĚR

Cílem mé bakalářské práce bylo nastudování problematiky obnovení dvou signálů z neúplného pozorování jejich křížového prolnutí. Dále navrhnout optimalizační algoritmus pro případ lineárního prolnutí a případ se šumem. Vše jsem měl implementovat v prostředí Matlab, ověřit funkčnost implementace a provést analýzu, za jakých podmínek lze daný typ úlohy řešit, případně s jakou chybou.
Tato práce uvádí velikost a vlastnosti digitálních obrázků, vytvoření křížového prolnutí dvou obrázků na základě konvexní kombinace a popis řešení obnovení těchto dvou obrázků. Dále je uveden způsob vytvoření neúplného křížového prolnutí dvou obrázků, jejich zašumění a znovu obnovení dvou původních obrázků. Podle zadání jsem měl vyřešit i problematiku pro nelineární prolnutí, ale z důvodu nedostatku času jsem jej nevypracoval.
Nakonec jsem uvedl praktické řešení v softwaru Matlab a výsledky obnovení obrázků z tohoto programovacího prostředí. V poslední kapitole jsou uvedeny chyby řešení a ukazuje se, že při lineárním křížovém prolnutí jsou v případě porovnání maximálních hodnot původních a obnovených obrázků nulové a při porovnání minimálních hodnotách téměř nulové. V případě zašuměných kombinací pro mnou zvolenou směrodatnou odchylku $\sigma = 30$ je řešení velmi uspokojivé, tedy obrázky jsou rozpoznatelné a zřetelně zbaveny šumu.
Podmínky řešení jsou takové, že problematika pro lineární prolnutí se dá řešit při splnění podmínek konvexní kombinace a obrázky musí mít stejné rozměry. To samé platí pro lineární prolnutí se šumem, navíc ale pro odšumění je potřeba použití minimálně prvních dvou hlavních komponent.
Od strany 28 nejsou v textu práce odstavce, což je způsobeno nepoužitím příkazu pro plovoucí prostředí na obrázky. Raději jsem totiž chtěl pro přehlednost a orientaci dát obrázky přímo pod odkazující text.

# LITERATURA


[1] KÁNSKÝ, A.: *Dvě úlohy z oblasti zpracování videa netradičními prostředky*: Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 35 s. Vedoucí práce byl doc. Mgr. Pavel Rajmic, Ph.D.

[2] SCHMIDTMAYER, J. *Maticový počet a jeho použití v technice*. 2. přeprac. vyd. Praha: SNTL, 1967. Teoretická knižnice inženýra. 382 s.

[3] BURGER, W.; BURGE, M. J.; *Digital Image Processing: An Algorithmic Introduction Using Java*, Springer-Verlag London, 2008. s. 11-13. ISBN 978-1-84628-379-6.

[4] OLŠÁK, P.: *Lineární algebra*. 2007, poslední aktualizace 31. 7. 2007 [cit. 12. 12. 2017]. s. 18. Dostupné z URL: <`http://petr.olsak.net/ftp/olsak/linal/linal.pdf`>

[5] RAJMIC, P.: *Lineární a konvexní kombinace signálů*. Dostupné z URL: <`https://www.youtube.com/watch?v=PgdDVZWhtSQ`>

[6] BALKOVÁ, L.: *Lineární algebra 1*. Dostupné z URL: <`https://kmlinux.fjfi.cvut.cz/~balkolub/Vyuka/zima2013/skriptaLA106_110KonvexniMnoziny.pdf`>

[7] *Wikipedie - internetová encyklopedie*: Normal distribution [online]. 2018 [cit. 18. 5. 2018]. Dostupné z URL: <`https://en.wikipedia.org/wiki/Normal_distribution`>

[8] TONHAUSEROVÁ, Z.: *Metoda hlavních komponent a její aplikace*: Diplomová práce. Olomouc: Univerzita Palackého v Olomouci, Přírodovědecká fakulta, Katedra matematické analýzy a aplikací matematiky, 2013. 68 s. Vedoucí diplomové práce byl Mgr. Ondřej Vencálek, Ph.D.

[9] SOLNICKÝ, J.: *Využití analýzy hlavních komponent (PCA) ke zpracování obrazových dat*: Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínského inženýrství, 2008. 48 s. Vedoucí práce byl Ing. Milan Rychtárik.

[10] *Wikipedie - internetová encyklopedie*: Principal Component Analysis [online]. 2018 [cit. 14. 5. 2018]. Dostupné z URL: <`https://en.wikipedia.org/wiki/Principal_component_analysis`>

[11] PCA. Dostupné z URL:
<https://www.mathworks.com/help/stats/pca.html>

[12] KALINA, J.; TEBBENS J.D.: *Metody pro redukci dimenze v mnohorozměrné statistice a jejich výpočet*. 2013 [cit. 13. 5. 2018] 17 s. Dostupné z URL:
<http://www.cs.cas.cz/duintjertebbens/pubs/KalinaDT.pdf>

[13] uint8. Dostupné z URL:
<https://www.mathworks.com/help/matlab/ref/uint8.html>

# SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

| | |
|---|---|
| $\mathbf{A}$ | obecná matice |
| $\alpha$ | koeficient pro násobení matic |
| $C$ | obecné označení kombinace |
| $\vec{m}$ | vektor m |
| $\mu$ | střední hodnota |
| $\mathbf{n}$ | matice šumu |
| PCA | analýza hlavních komponent |
| $\sigma$ | směrodatná odchylka |
| SNR | odstup signálu od šumu |
| SVD | singulární rozklad |
| $\vec{t}$ | vektor t |
| $\vec{v}$ | obecný vektor v |

# SEZNAM PŘÍLOH

# A OBSAH PŘILOŽENÉHO CD

## A.1 Programy

Na CD jsou přiloženy všechny potřebné soubory pro předvedení výsledků řešení. Dále jsem vytvořil dva textové dokumenty s popisem návodu na postupné spuštění jednotlivých skriptů pro případ bez šumu a s šumem.
Kódy byly testovány ve verzi Matlab 2014b.

## A.2 Použité obrazy

ostruzina
pampeliska