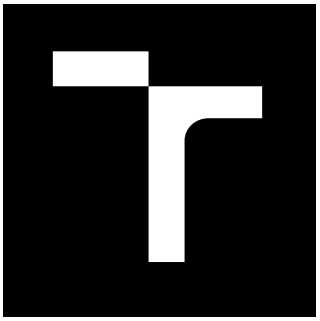


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

DETEKCE PERIODICKÝCH TVARŮ V OBRAZE

DETECTION OF PERIODIC SHAPES IN AN IMAGE

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Michal Kasala**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Dominik Řičánek**

SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Michal Kasala ***ID:*** 247393

Ročník: 3 ***Akademický rok:*** 2024/25

NÁZEV TÉMATU:

Detekce periodických tvarů v obraze

POKYNY PRO VYPRACOVÁNÍ:

Úkolem studenta bude implementace algoritmu schopného najít v obraze libovolný periodický signál. Student dostane videa obsahující vibrující objekt, ze kterých si následně bude vytvářet „timeslice“ obrazy. Na těchto obrazech pak bude mít za úkol nalézt a parametrizovat periodický signál, který bude většinou v podobě sinusové vlny o jedné frekvenci. Jako bonus budu považovat schopnost nalezení signálů s útlumem nebo složený z vícero frekvencí.

1. Proveďte rešerši analýzy periodických signálů a hledání tvarů v obraze (Fourierova transformace, Houghova transformace).
2. Implementujte algoritmus pro nalezení a parametrizaci periodického signálu v obraze. Minimálně signálu s konstantní amplitudou a jednou frekvenční složkou.
3. Otestujte správné fungování algoritmu a přesnost parametrizace vašeho algoritmu.

DOPORUČENÁ LITERATURA:

PRESS, William H. Numerical recipes: the art of scientific computing. 3rd ed. Cambridge: Cambridge University Press, 2007. ISBN 978-0-521-88407-5.

SOJKA, Eduard. Digitální zpracování a analýza obrazů. Ostrava: VŠB - Technická univerzita, 2000. ISBN 80-7078-746-5.

Termín zadání: 10.2.2025 ***Termín odevzdání:*** 28.5.2025

Vedoucí práce: Ing. Dominik Řičánek

Ing. Miroslav Jirgl, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato práce se zabývá detekcí a odhadem parametrů sinusoidních vzorů v obrazech reprezentujících periodický pohyb. Cílem je navrhnout a implementovat algoritmus, který v obraze lokalizuje sinusoidní trajektorii a určí její frekvenci, amplitudu a orientaci. Navržený postup využívá prostorové analytické metody, jako je detekce extrémů a analýza gradientu. Algoritmus je testován výhradně na synteticky generovaných datech s předem známými parametry. V závěru jsou diskutována omezení navrženého řešení a možnosti jeho rozšíření.

KLÍČOVÁ SLOVA

Detekce sinusoidních vzorů, zpracování obrazu, analýza gradientu, odhad parametrů, syntetická data, detekce extrémů, Houghova transformace, obecná Houghova transformace, segmentace obrazu

ABSTRACT

The thesis deals with the detection and parameter estimation of sinusoidal patterns in images representing periodic motion. The main goal is to design and implement an algorithm that localizes the sinusoidal trajectory in the image and determines its frequency, amplitude, and orientation. The proposed method combines spatial analysis techniques, including peak detection and gradient evaluation. The algorithm is tested exclusively on synthetically generated data with known parameters. Limitations and potential improvements are discussed in the conclusion.

KEYWORDS

Sinusoidal pattern detection, image processing, gradient analysis, parameter estimation, synthetic data, peak detection, Hough transform, Generalized Hough transform, image segmentation

KASALA, Michal. *Detekce periodických tvarů v obraze*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2025. Vedoucí práce: Ing. Dominik Řičánek

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Michal Kasala
VUT ID autora:	247393
Typ práce:	Bakalářská práce
Akademický rok:	2024/25
Téma závěrečné práce:	Detekce periodických tvarů v obraze

Prohlašuji, že svou závěrečnou práci jsem vypracoval 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 autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom 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*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Dominiku Řičánkovi, za odborné vedení, konzultace, trpělivost, shovívavost, podnětné návrhy k práci a především velkou vstřícnost.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Zpracování obrazových dat často zahrnuje detekci a analýzu opakujících se vzorů, mezi které patří i periodické průběhy, například sinusoidy. Tyto tvary se v technické praxi objevují v mnoha oblastech – od vibrační analýzy po optické interferenční jevy. Cílem této bakalářské práce je navrhnout a implementovat algoritmus, který bude schopen v obraze rozpoznat periodický sinusový průběh a určit jeho základní parametry, jako jsou frekvence, amplituda a směr natočení.

V práci jsou nejprve představeny vybrané teoretické nástroje využitelné pro detekci takových vzorů, zejména Houghova transformace a principy frekvenční analýzy. Hlavní důraz je však kladen na praktickou realizaci detekčního algoritmu. V rámci řešení byly vytvořeny syntetické obrazové vstupy obsahující sinusoidy s různým zkreslením a šumem. Celý algoritmus byl implementován v prostředí MATLAB, které poskytuje vhodné nástroje pro práci s obrazovými daty a numerické výpočty.

Navržený postup je otestován na připravených datech a jeho výstupem jsou odhadnuté parametry průběhu. Součástí práce je i porovnání několika přístupů ke zpracování a diskuse nad jejich výhodami a limity v kontextu analyzovaných dat.

1 Zpracování signálů

Signál lze obecně definovat jako fyzikální veličinu, která nese informaci a může se měnit v čase nebo prostoru. Signály jsou základním nositelem dat ve většině technických i přírodních systémů. Typickými příklady jsou akustické vlny (zvuk), elektrické napětí, elektromagnetické vlny nebo například vibrace mechanických těles.

Z pohledu zpracování se signály dělí na:

- **Spojité signály:** Jsou definovány pro každou hodnotu času v $\mathbb{R}$. Příkladem může být sinusový průběh napětí ve střídavé síti. Tyto signály mají nekonečně mnoho hodnot v konečném časovém intervalu, a proto je nelze přímo uložit ani zpracovávat na číslicovém zařízení.
- **Diskrétní signály:** Jsou definovány pouze v určitých časových okamžicích, tzv. vzorcích. Vznikají vzorkováním spojitého signálu, přičemž výsledný signál je reprezentován konečnou posloupností hodnot vhodnou pro digitální zpracování.

1.1 Vzorkování signálů

Digitální zpracování signálů vyžaduje převod spojitého signálu, který má nekonečné množství hodnot v čase, na signál diskrétní. Uchování každého okamžiku spojitého signálu by v praxi znamenalo nutnost nekonečné paměti a neomezeného výpočetního výkonu, což není technicky realizovatelné. Proto je nutné původní spojitý signál převést do reprezentace, kterou lze efektivně uložit a dále analyzovat či upravovat. Tento proces se nazývá vzorkování.

Vzorkování je postup, při kterém se ze spojitého signálu *x(t)* odebírají hodnoty v pravidelných časových intervalech, čímž vzniká posloupnost diskrétních hodnot *x(n)*, definovaná vztahem:

$$x(n) = x(n \cdot T_s), \qquad n = 0, 1, 2, \ldots, N-1 \tag{1.1}$$

Zde T_s označuje vzorkovací periodu, tedy časový interval mezi dvěma po sobě jdoucími vzorky. Tomu odpovídá vzorkovací frekvence f_s, která udává, kolik vzorků je odebráno za jednu sekundu. Obě veličiny spolu souvisí inverzním vztahem:

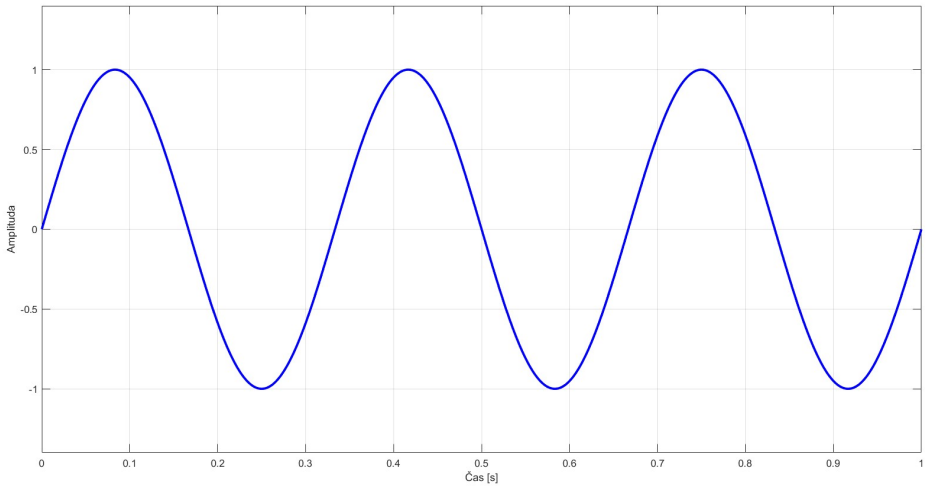
$$f_s = \frac{1}{T_s} \quad [\text{Hz}] \tag{1.2}$$

$$T_s = \frac{1}{f_s} \quad [\text{s}] \tag{1.3}$$

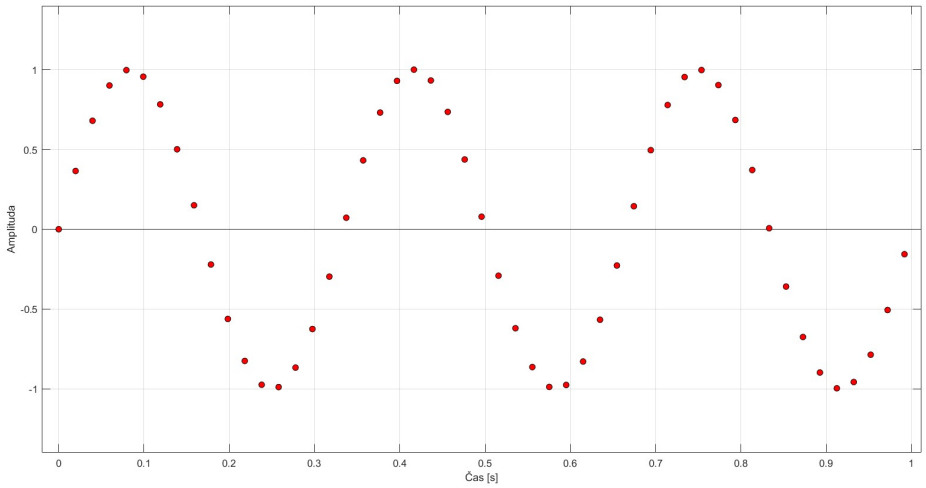
Správná volba vzorkovací frekvence je klíčová pro zachování informací obsažených ve spojitém signálu. Pokud se signál vzorkuje příliš pomalu, dochází ke zkreslení zvanému aliasing, které znemožňuje správnou rekonstrukci původního signálu. Tomuto tématu a související problematice se věnuje následující podkapitola.

1.1.1 Rekonstrukce signálu

Mějme daný signál o jediné frekvenci, který chceme navzorkovat a následně zpětně zrekonstruovat.

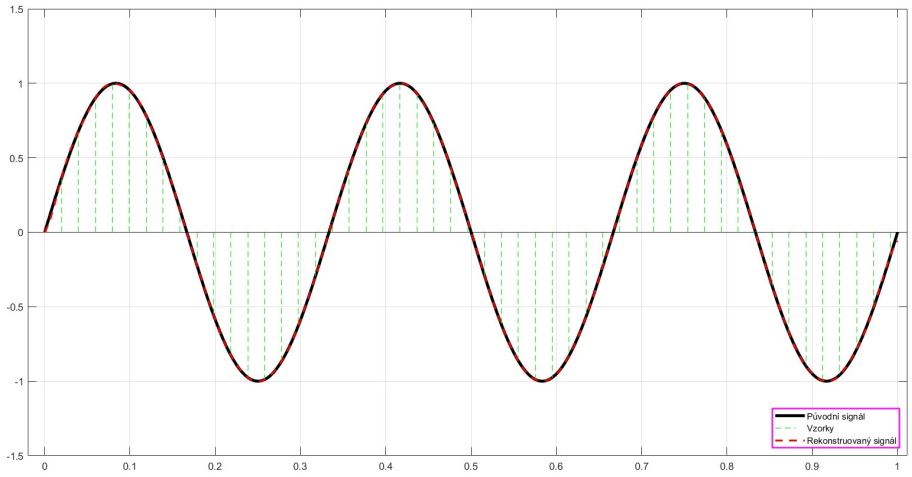


Obr. 1.1: Spojitý signál o frekvenci 3 Hz



Obr. 1.2: Diskrétní signál vzniklý vzorkováním spojitého

Pomocí interpolace můžeme vzít takto diskrétní signál a získat z něj zpět původní spojitý.



Obr. 1.3: Zpětná rekonstrukce spojitého signálu

1.1.2 Nyquistův–Shannonův vzorkovací teorém

Tento teorém popisuje, jakým způsobem by se měl vzorkovat signál, bez toho, aniž by se z něj ztratila jakákoliv informace.

Představme si, že máme frekvenčně ohraničený signál $X(t)$, to znamená, že když bychom chtěli udělat Fourierovu transformaci z daného signálu, $\hat{X}(f) = \mathcal{F}\{X(t)\}$, narazili bychom na jistou frekvenci f_{max} pro kterou platí:

$$|\hat{X}(f)| = 0 \quad \forall\, |f| > f_{max} \tag{1.4}$$

tedy signál nepřenáší žádnou energii nad maximální frekvencí f_{max}.

Teorém dále říká, že pro správné zachování signálu ho musíme vzorkovat frekvencí alespoň dvakrát větší než největší frekvence v něm obsažená:

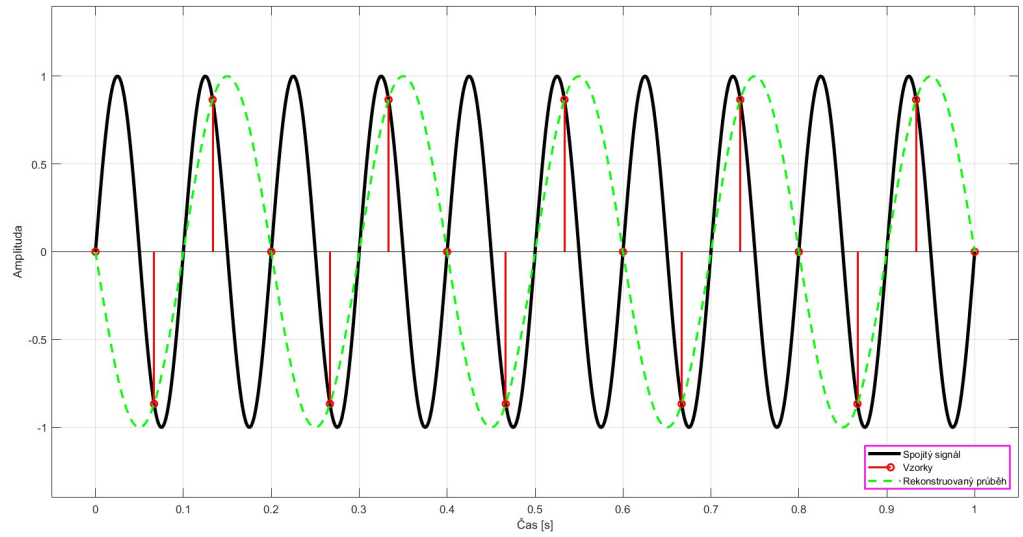
$$f_s \geq 2 \cdot f_{max} \tag{1.5}$$

Pokud naše vzorkovací frekvence splňuje tuto podmínku, měli bychom být schopni přesně rekonstruovat původní signál ze signálu vzorkovaného [1].

1.1.3 Aliasing

Tento jev nastává, když nedodržíme podmínku Nyquistova–Shannonova vzorkovacího teorému. Tedy vzorkujeme signál frekvencí $\boldsymbol{f}$, která je **nižší** než $2 \cdot f_{max}$. Pojďme si názorně ukázat, jak to vypadá v praxi.

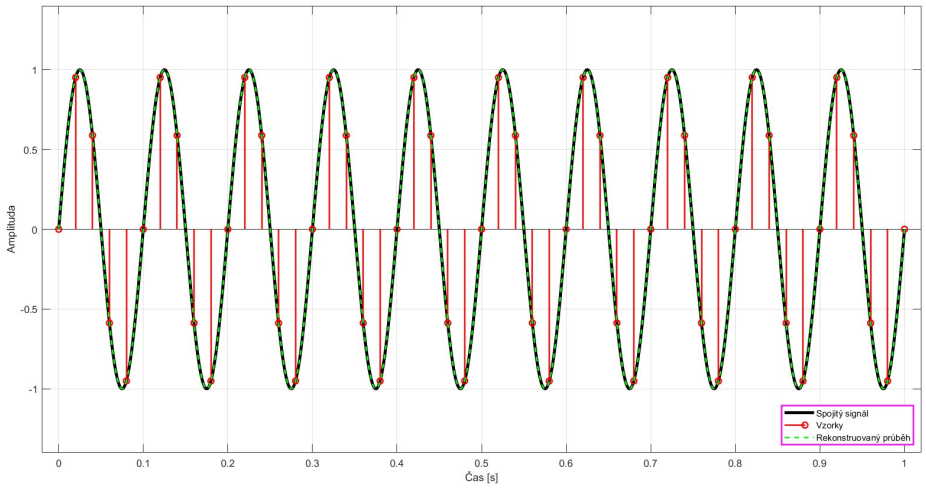
Na grafu 1.4 je zobrazen signál o frekvenci 10 Hz, který byl vzorkován frekvencí $f_s = 15$ Hz.



Obr. 1.4: Vznik aliasingu jako důsledek malé vzorkovací frekvence

Jak je vidět, tak rekonstruovaný signál není shodný s původním signálem (v našem případě má poloviční frekvenci).

Aliasing je třeba ošetřit při každém vzorkování jakéhokoliv signálu, jinak nebude možné správně reprezentovat původní signál a to povede k nesprávným závěrům měření.



Obr. 1.5: Signál s dostatečně velkou vzorkovací frekvencí

1.1.4 Antialiasingový filtr

Před vzorkováním je běžné aplikovat antialiasingový filtr, což je dolní propust (low-pass filtr), která odstraní složky signálu s frekvencí vyšší než Nyquistova frekvence f_N. Tím se předejde aliasingu.

1.1.5 Nyquistova frekvence

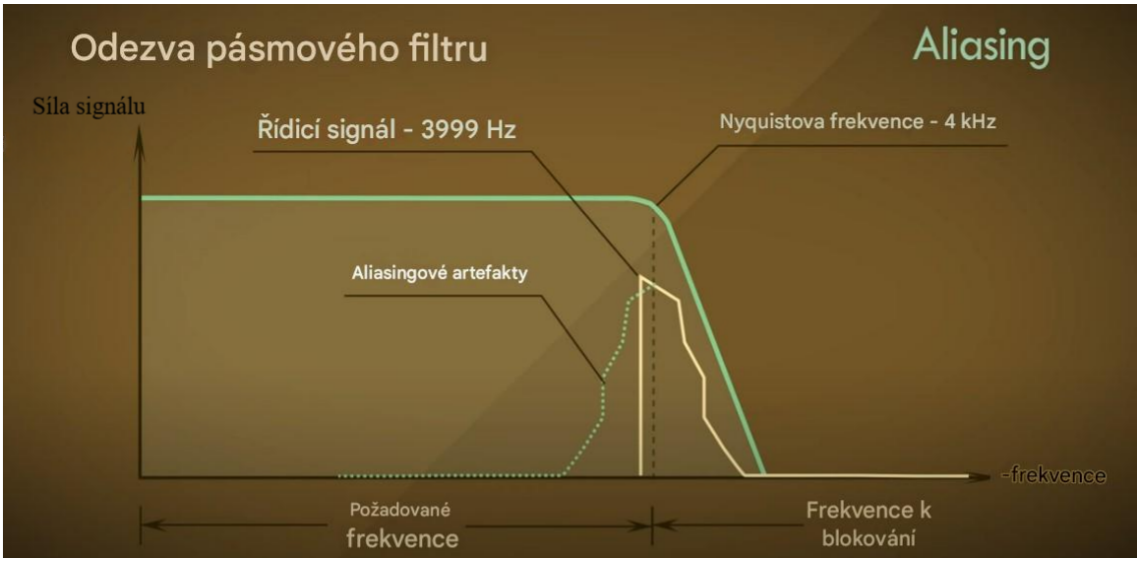
Nyquistova frekvence f_N je definována jako polovina vzorkovací frekvence f_s

$$f_N = \frac{f_s}{2} \quad [Hz] \tag{1.6}$$

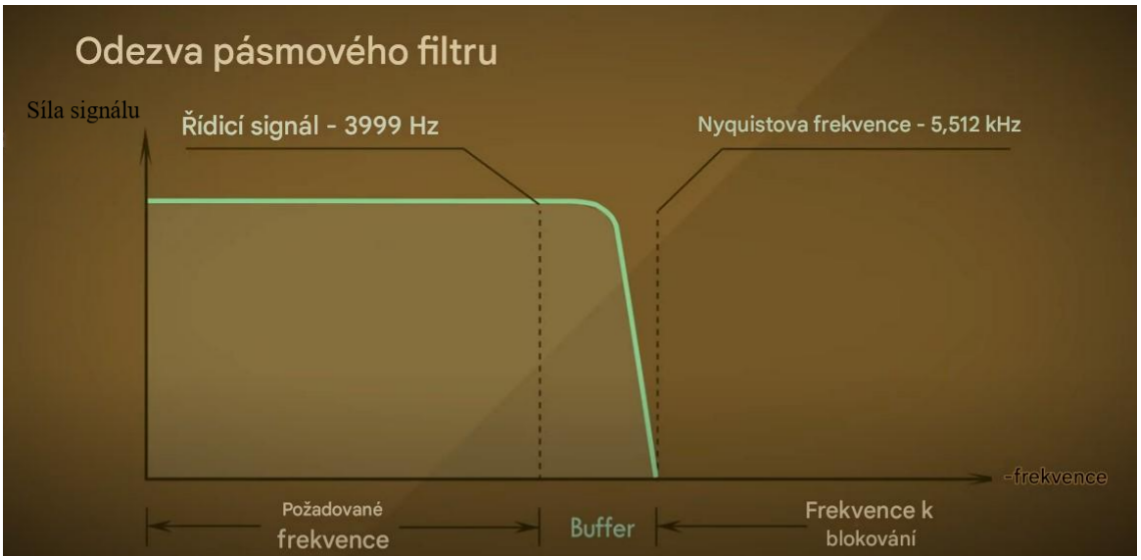
Podle Nyquistova–Shannonova vzorkovacího teorému musí být vzorkovací frekvence f_s alespoň dvojnásobná oproti maximální frekvenci signálu f_{max}, aby nedošlo ke zkreslení (aliasingu). V praxi se však často používá ještě větší rezerva, například:

$$f_s > 2,5 \cdot f_{max}$$

Důvodem je omezená rychlost reakce reálných hardwarových filtrů, které nedokáží ideálně a okamžitě potlačit nežádoucí frekvence. Místo toho frekvence rychle, ale postupně slábnou, což vyžaduje vyšší vzorkovací frekvenci. Tento efekt si můžeme také ilustrovat na následujícím obrázku:

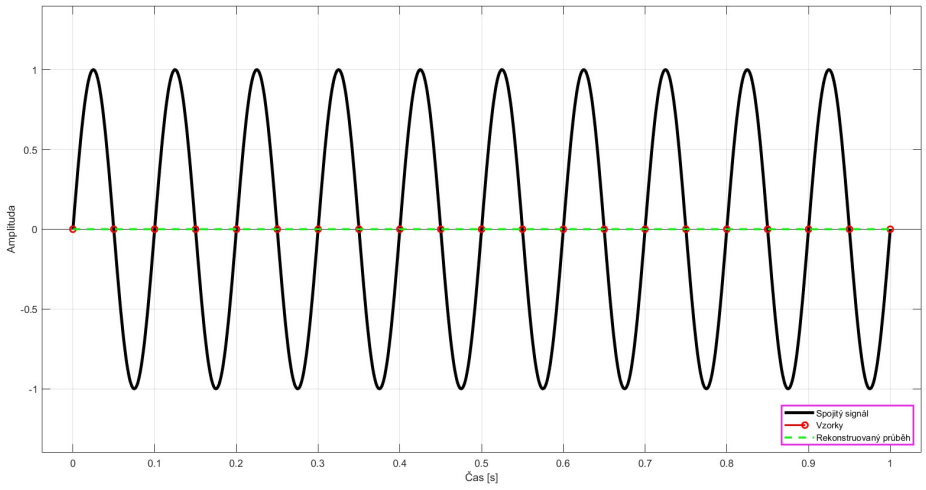


Obr. 1.6: Filtr propouští nechtěné frekvence - aliasing [2]



Obr. 1.7: Filtr má dostatečnou rezervu pro útlum frekvencí [2]

Co se stane, pokud vzorkovací frekvence f_s bude přesně dvojnásobkem nejvyšší frekvence signálu f_{max}? Tento případ ilustruje následující graf:



Obr. 1.8: Vzorkovací frekvence rovna dvojnásobku Nyquistovy

Signál je vzorkován přesně v okamžicích, kdy prochází nulovou hodnotou (osy x). Výsledný vzorkovaný signál proto vypadá jako nulový průběh, což může vést k mylnému závěru, že signál vůbec neexistuje.

2 Fourierova transformace

Za jejím vznikem stojí francouzský matematik Jean-Baptiste Joseph Fourier, jenž představil svůj návrh v knize *Théorie analytique de la chaleur* (Analytická teorie tepla).

2.1 Definice

Jedná se o nástroj, který nám umožňuje dekompozici (rozklad na jednodušší části) funkce. Jako vstupní parametr se zadá funkce, která:

a) je absoluně integrovatelná
b) má konečný počet nespojitostí [5]

V literatuře se objevují různé podmínky, které by funkce měla splňovat (stejně tak i odlišné formulace Fourierova integrálu) a zdá se, že se na nich matematici neshodnou. Proto jsem vybral 2, které se objevují nejčastěji a zároveň jsou i nejméně přísné.

Výstupem fourierovy transformace (FT) je komplexní spektrum skládající se z amplitudové a fázové složky, což popisuje vztah:

$$\mathcal{F}(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft}\, dt \tag{2.1}$$

kde

j	imaginární jednotka, j $= \sqrt{-1}$
e^{jx}	Eulerův vzorec, $e^{jx} = cos(x) + j \cdot sin(x)$

2.2 Základní vlastnosti

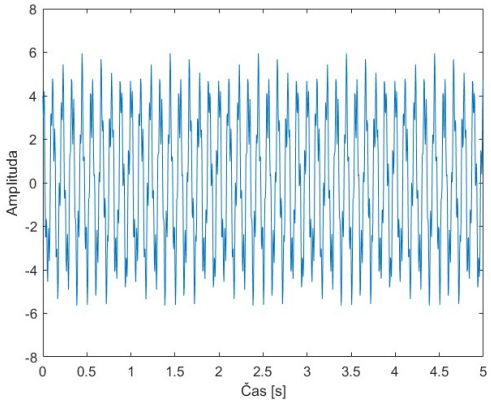
FT disponuje několika příhodnými vlastnostmi:

Tab. 2.1: Vybrané vlastnosti Fourierovy transformace [6] [7]

Vlastnost	Časová doména	Frekvenční doména
Linearita	$\alpha f_1(x) + \beta f_2(x)$	$\alpha \mathbb{F}_1(x) + \beta \mathbb{F}_2(x)$
Škálování v čase	$f(at)$	$\frac{1}{\lvert a \rvert}\mathbb{F}\left(\frac{u}{a}\right)$
Škálování ve frekvenci	$\frac{1}{\lvert b \rvert} f\left(\frac{t}{b}\right)$	$\mathbb{F}(bu)$
Posun v čase	$f(t - t_0)$	$\mathbb{F}(u)e^{j2\pi ft_0}$
Posun ve frekvenci	$f(t)e^{-j2\pi f_0 t}$	$\mathbb{F}(u - u_0)$

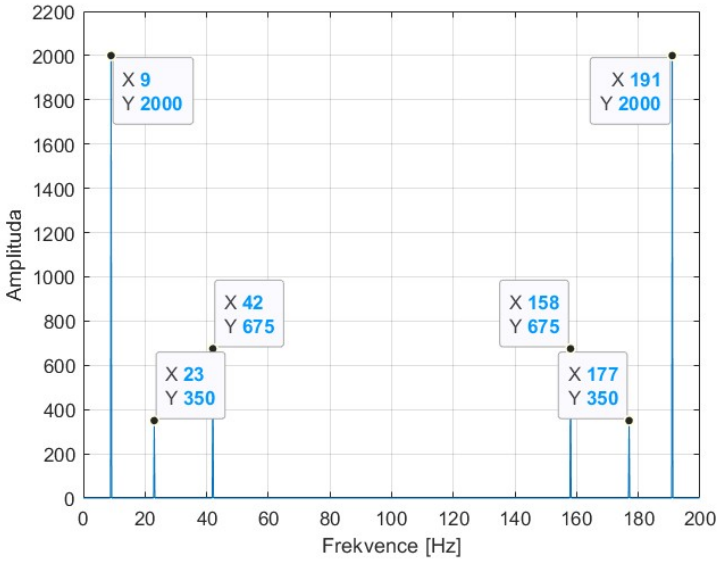
2.3 Analýza jednoduchého signálu

Představme si, že nám někdo předloží následující graf s tím, že máme najít frekvenci a amplitudu sinusoidů, jež ho tvoří.



Obr. 2.1: Předložený signál

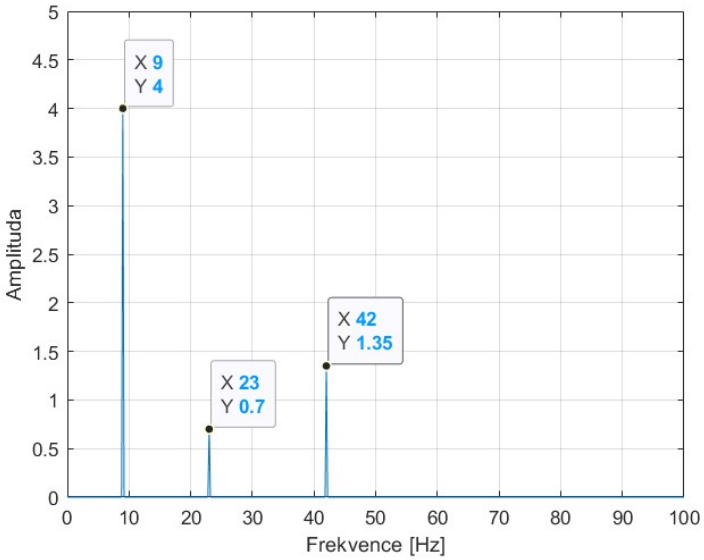
Přirozeně jako první použijeme FFT a dostaneme frekvenční spektrum.



Obr. 2.2: Dvoustranné frekvenční spektrum

Vidíme, že je tvořeno 6 vrcholy. Jedná se totiž o dvoustranné spektrum, kde je první polovina hodnot tvořena kladnými frekvencemi a druhá zápornými. Vzájemně se zrcadlí okolo Nyquistovy frekvence, která leží přesně uprostřed. Stejnosměrná složka pak leží na frekvenci 0 Hz.

Nyní už stačí jen vykreslit jednu stranu (kladnou), vynásobit spektrum dvěma pro kompenzaci odebrání poloviny hodnot a podělit spektrum počtem vzorků.



Obr. 2.3: Jednostranné frekvenční spektrum

Tak tedy náš zadaný signál obsahuje celkem 3 sinusoidy: první o frekvenci 9 Hz s amplitudou 4, druhá 23 Hz s amplitudou 0.7 a třetí 42 Hz s amplitudou 1.35.

2.3.1 Inverzní Fourierova transformace

Inverzní Fourierova transformace (IFT) převádí frekvenční spektrum zpět na časovou oblast, to popisuje rovnice:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \mathcal{F}(f) \cdot e^{j2\pi ft}\, df \tag{2.2}$$

2.3.2 Diskrétní Fourierova transformace

Diskrétní Fourierova transformace neboli DFT se používá pro analýzu diskrétních signálů, které se získají vzorkováním jejich spojitých verzí.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-j\frac{2\pi kn}{N}} \tag{2.3}$$

2.3.3 Inverzní diskrétní Fourierova transformace

Neboli IDFT, převádí frekvenční spektrum zpět na časovou oblast, to popisuje rovnice:

$$x_n = \frac{1}{N} \sum_{n=0}^{N-1} X_k \cdot e^{j\frac{2\pi kn}{N}} \tag{2.4}$$

2.3.4 Rychlá Fourierova transformace

Algoritmus rychlé Fourierovy transformace (FFT) je efektivní metodou pro výpočet diskrétní Fourierovy transformace (DFT), která převádí signál z časové nebo prostorové domény do domény frekvenční. Ve zpracování obrazu se využívá 2D varianta FFT, která se aplikuje na dvojrozměrná data, tedy na samotné obrazové matice.

Základem FFT je princip rozděl a panuj (divide and conquer), který umožňuje výrazně zrychlit výpočty oproti přímému použití DFT. Tento princip funguje tak, že se původní problém rozdělí na několik menších podproblémů, které jsou jednodušší na výpočet. Tyto podproblémy se dále rekurzivně dělí, dokud není možné výpočty provést přímo. Poté se výsledky postupně skládají zpět do celkového řešení.

V případě jednorozměrné FFT se vstupní sekvence rozdělí na sudé a liché indexy a algoritmus se rekurzivně aplikuje na tyto části. Tento postup lze přímo zobecnit i pro dvourozměrná data. Pro výpočet 2D FFT se nejprve provádí FFT podél řádků obrazu a následně FFT podél sloupců (nebo naopak). Tímto způsobem se 2D problém rozdělí na množství jednodušších 1D problémů, což je výpočetně velmi efektivní.

Díky rekurzivnímu rozdělování a spojování výpočtů umožňuje FFT snížit časovou složitost z původních $O(N^2)$ operací na přibližně $O(N \cdot log(N))$, což je zvláště výhodné při zpracování velkých obrazových dat. FFT se tak často používá v oblasti frekvenční filtrace, analýzy textury, spektrální analýzy nebo detekce periodických struktur v obraze.

3 Zpracování dvourozměrného signálu

Jednou z významných aplikací Fourierovy transformace je zpracování dvourozměrných signálů, tedy obrazů. Analýza ve frekvenční doméně přináší jiný pohled na strukturu obrazových dat a umožňuje například efektivní filtraci nebo detekci hran. Značného uplatnění dosahuje také v oblasti komprese obrazu, zejména při převodu do formátu JPEG, kde se využívá modifikovaná forma transformace k potlačení vizuálně nevýznamných frekvenčních složek.

Dvourozměrná diskrétní Fourierova transformace je definována rovnicí:

$$\mathbf{F}(u,v) = \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} f(x,y) \cdot e^{-2\pi j[\frac{xu}{N}+\frac{yv}{M}]} \tag{3.1}$$

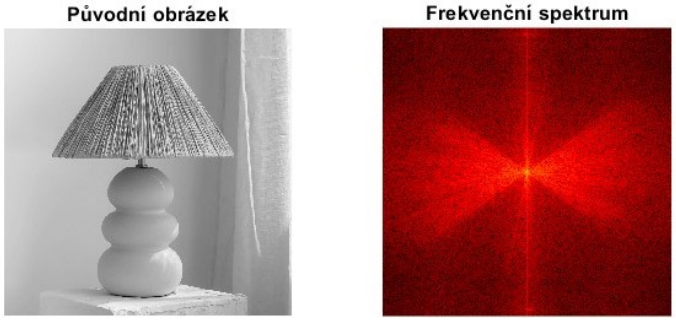
Následující ukázka demonstruje využití frekvenčního zpracování obrazu na konkrétním příkladu běžného předmětu. Cílem je aplikovat filtraci pomocí vhodně zvoleného frekvenčního filtru a následně porovnat výsledky s původním obrazem.

Obr. 3.1: Obrázek pro filtraci [8]

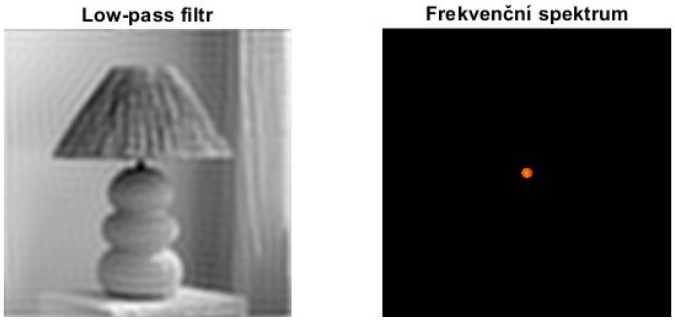
Nejprve je obraz převeden do odstínů šedi. Následně je na něj aplikována dvourozměrná Fourierova transformace. V případě vícerozměrných signálů se obvykle postupuje tak, že se transformace nejprve provede po řádcích a následně po sloupcích (nebo naopak). Díky komutativní vlastnosti Fourierovy transformace nezáleží na pořadí těchto operací.

Výsledné spektrum je posunuto tak, aby bylo symetrické vůči počátku souřadnicového systému. Díky tomu jsou nízké frekvence zobrazeny ve středu obrazu a vysoké frekvence se nacházejí směrem k okrajům. Pro lepší vizualizaci je spektrum vykresleno s použitím barevné mapy typu hot, kde světlejší barvy reprezentují vyšší intenzitu (dominantní frekvence).

Pro ilustraci vlivu frekvenční filtrace byl následně aplikován dolní propustní filtr (low-pass filter), jehož cílem je potlačit vysokofrekvenční složky obrazu.



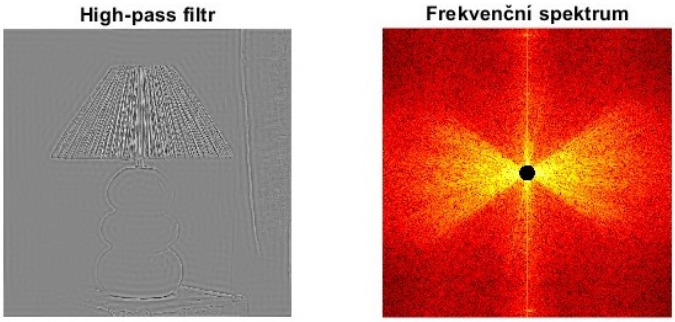
Obr. 3.2: Frekvenční spektrum obrazu lampy



Obr. 3.3: Aplikace low-pass filtru ve frekvenční doméně

Výsledkem této úpravy je zjemnění detailů, což se ve výsledném obraze projeví jako mírné rozmazání. Tento efekt je očekávaný, neboť vysokofrekvenční složky odpovídají jemným strukturám a hranám v obraze, které byly filtrem odstraněny.

V dalším kroku bude pro srovnání aplikován filtr horní propustnosti (high-pass filter), který naopak potlačí nízké frekvence a ponechá pouze vysoké. Takový filtr by měl zvýraznit detaily a hrany přítomné v původním obrazu.

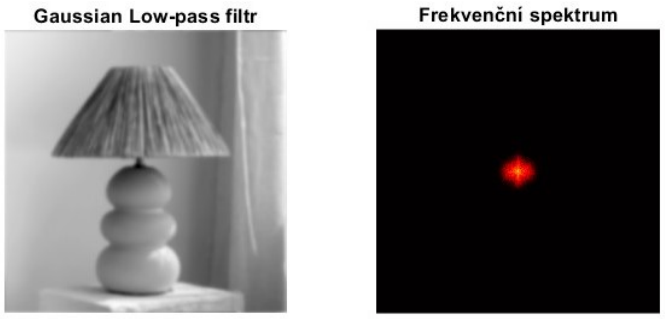


Obr. 3.4: Spektrum high-pass filtru

Při pohledu na výsledné obrázky po aplikaci filtrů si lze všimnout jemných oscilací v okolí výrazných hran. Tento jev se objevuje především u jednoduše konstruovaných filtrů, které razantně potlačí určité frekvenční složky. Právě tento způsob

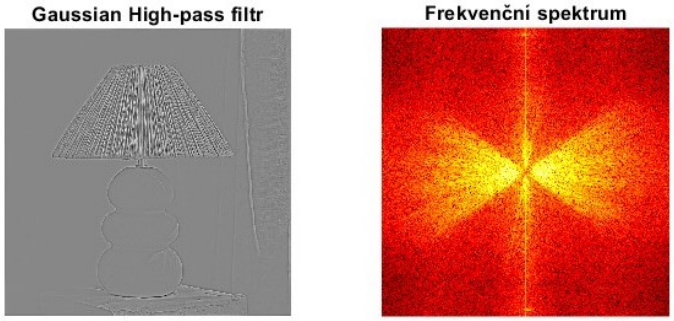
ořezu spektra může vést k takovým vizuálním artefaktům.

Pro minimalizaci artefaktů se využívají různé metodiky. Já jsem použil Gaussův filtr, což je jeden ze skupiny vyhlazovacích filtrů. Po jeho aplikaci by se tedy měla kvalita obrázku zlepšit.

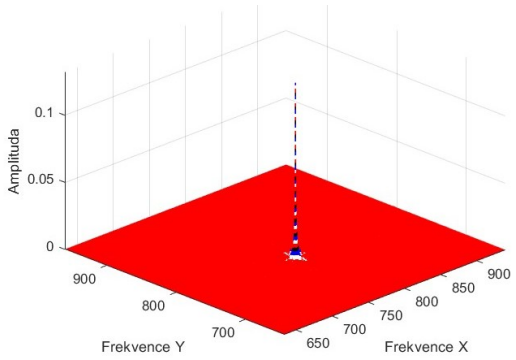


Obr. 3.5: Gaussův low-pass filtr

Je to způsobeno právě tím, že frekvenční spektrum už není ostrý kruh, ale postupně klesající vrchol. A vidíme, že i pro high-pass filtr vyhlazení funguje.



Obr. 3.6: Gaussův high-pass filtr



Obr. 3.7: Amplitudové spektrum ve 3D

Na obr. 3.7 se dá krásně ukázat, proč můžeme odstranit velkou část vysokých frekvencí a stále získáme obrázek v dobré kvalitě. Naprostá většina dominantních frekvencí se nachází blízko středu a zbytek má zanedbatelnou amplitudu.

4 Houghova transformace

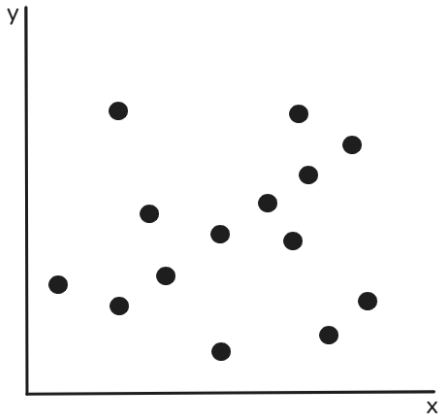
Ve zpracování obrazu hraje klíčovou roli identifikace geometrických útvarů, jako jsou přímky, kružnice nebo elipsy. Tyto tvary často představují významné příznaky (features) při rozpoznávání objektů, analýze scén nebo při měření v technických aplikacích. Obzvlášť v případech, kdy máme k dispozici pouze binární mapu hran, bez dalších kontextových informací, je robustní metoda pro detekci těchto tvarů nezbytná.

Houghova transformace je klasický algoritmus, který umožňuje detekovat parametry geometrických útvarů i v případě, že jsou částečně neúplné, překryté nebo zasažené šumem. Byla patentována Paulem Houghem v roce 1962 a původně sloužila k detekci přímek. Postupem času však byla rozšířena i na detekci dalších tvarů, jako jsou kružnice a elipsy, čímž se z ní stal univerzální nástroj pro identifikaci analyticky popsatelných útvarů v obraze.

4.1 Princip

Houghova transformace (HT) využívá základních poznatků o tvarech, které chceme detekovat. Nedělá nic jiného, než že vezme rovnici dané křivky a převede ji na rovnici parametrů. Pro každý bod na hraně v obrazovém prostoru pak spočítá, jaké parametry přímky by jím mohly procházet, a tyto informace zaznamená do tzv. akumulátoru – vícerozměrného pole, ve kterém každý prvek reprezentuje určitý tvar (například přímku s konkrétními parametry). Postupným přičítáním hlasů od jednotlivých bodů akumulátor „odhaluje“ nejpravděpodobnější přímky přítomné v obraze.

Představme si, že jsme dostali za úkol nalézt přímku, která se nachází v obrázku 4.1:



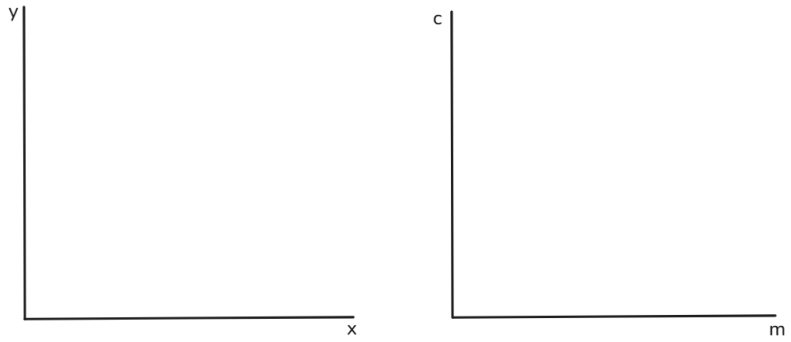
Obr. 4.1: Detekované body

Jako první, co uděláme, je, že si upravíme rovnici přímky:

$$\mathbf{y} = m \cdot \mathbf{x} + c \tag{4.1}$$

do rovnice parametrů:

$$\mathbf{c} = -\mathbf{m} \cdot x + y \tag{4.2}$$

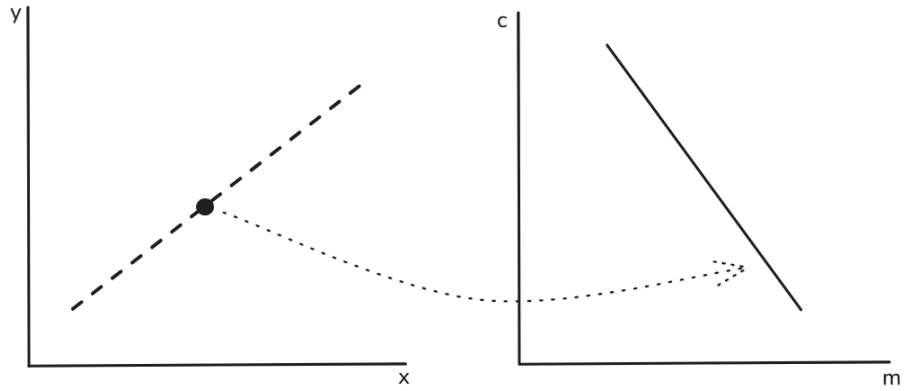


Obr. 4.2: Levá strana - rovina obrázku, pravá strana - rovina parametrů

Převedením rovnice přímky do tvaru c = - mx + y jsme vytvořili vztah mezi parametry m a c pro daný bod (x,y). Místo toho, abychom v obraze hledali přímku jako množinu bodů, teď pro každý bod hledáme množinu přímek, které by jím mohly procházet. Tento přístup nám umožňuje systematicky prohledávat prostor možných přímek a později vybrat ty, které odpovídají skutečné struktuře v obraze.

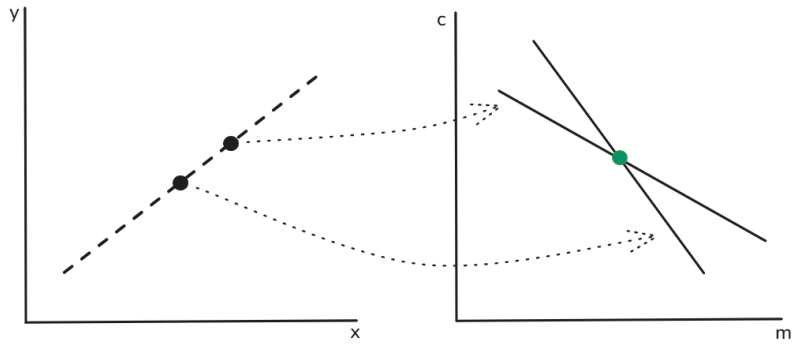
Nyní si představme, že máme vyhlídnutou určitou přímku, která propojuje naše body. Prozatím dáme stranou samotnou přímku a budeme se teď chvíli soustředit na body, které protíná.

Jako první si představíme jeden bod. Když vezneme jeho souřadnice $[x_i\ y_i]$ a dosadíme je do rovnice 4.2, výsledek bude přímka v rovině parametrů.



Obr. 4.3: Houghova transformace bodu v přímku

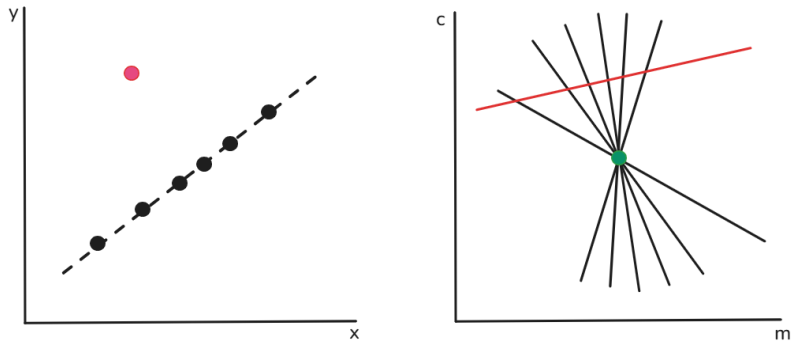
Následně přidáme druhý bod, který se opět transformuje na přímku.



Obr. 4.4: Houghova transformace bodů v přímky

Vidíme, že vzniklé přímky se nám protnou v určitém bodě. Je to právě tento bod, který nám reprezentuje detekovanou přímku v rovině obrázku. Čím více bodů budeme mít, tím výrazněji se nám odhalí přítomnost přímky v obrázku.

Ještě si můžeme ukázat, jak se projeví bod, který **neleží** na námi vyhlídnuté přímce.



Obr. 4.5: Nalezení přímky

I tento bod vytvoří přímku v rovině parametrů, ale má již odlišný směr a polohu.

K naší smůle není vhodné použití rovnice 4.2 pro HT, protože přímky nejsou nijak ohraničeny a pro jejich strmost tak připadá celá $\mathbb{R}$:

$$-\infty \leq m \leq \infty$$

Za předpokladu, že bychom chtěli zkoušet co největší množství hodnot strmosti m, vyústilo by to v enormní akumulátor a velkou výpočetní náročnost.

Z toho důvodu se u HT standardně používá tzv. normální parametrizace:

$$\rho = x \cdot cos(\theta) + y \cdot sin(\theta) \tag{4.3}$$

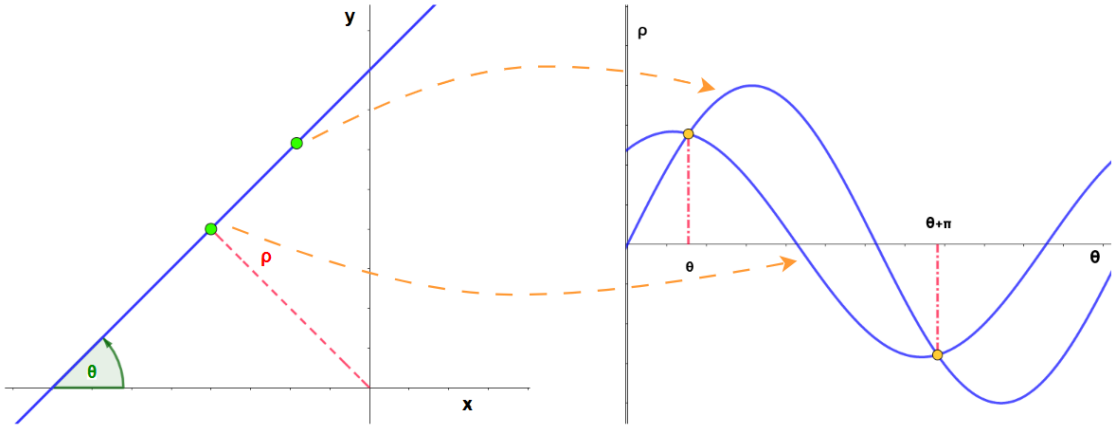
kde

- ρ vzdálenost mezi počátkem a přímkou
- θ úhel mezi osou x a přímkou

Když ještě omezíme úhel θ v rozmezí:

$$0 \leq \theta < \pi$$

dostáváme parametrizaci, se kterou se dá pěkně pracovat, protože teď už máme oba parametry omezené. Jeden úhlem 180° a druhý je omezen samotnými rozměry obrázku. Opět si vše názorně ukážeme pomocí grafů, ale ještě před tím si musíme uvědomit, že naše nově přepsaná rovnice už nebude generovat v Houghově prostoru přímky, nýbrž sinusoidy.



Obr. 4.6: Transformace přímky na sinusoidu

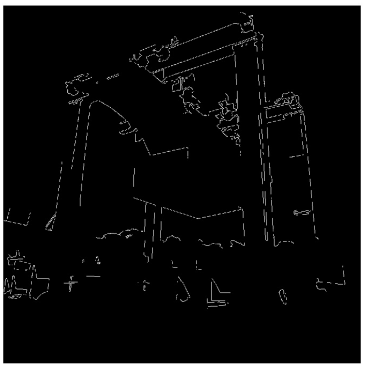
Průsečíky těchto sinusoid reprezentují body, které budou ležet na stejné přímce.

4.1.1 Ukázka detekce přímek v Matlabu

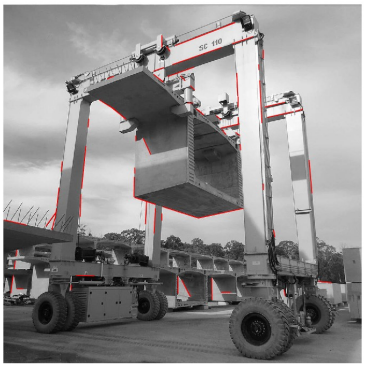
Pro tuto ukázku jsem si našel obrázek mobilního jeřábu. Jako první jsem jej převedl na stupně šedi. Následně prahoval pomocí zabudované funkce edge, kde jsem použil Cannyho detektor.

Obr. 4.7: Parametrická rovina přímek jeřábu - obraz byl otočen o 90°

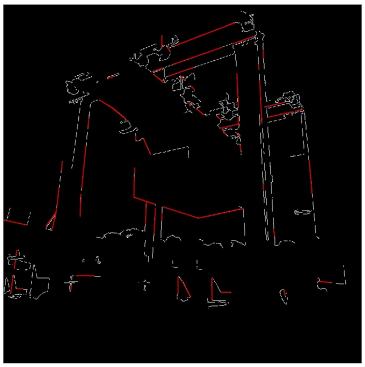
(a) Ukázkový snímek - jeřáb[4]



(b) Prahovaný jeřáb



(c) Detekované čáry na šedém obraze



(d) Detekované čáry na prahovaném obraze

Obr. 4.8: Jednoduchá detekce přímek v prostředí Matlab

4.2 Detekce kružnic

Přestože kružnice jsou jednodušší převést do parametrického prostoru než přímky díky rovnici, která je definuje. Zde se k naší smůle začíná projevovat nevýhoda použití HT. Jedná se o nevýhodu výpočetní.

Rovnice kruhu:

$$(x-a)^2 + (y-b)^2 = r^2 \tag{4.4}$$

V rovnici 4.4 se nachází tři parametry a, b a r. Kde a společně s b udávají střed kružnice ve směru osy x a osy y. Parametr r udává její poloměr. V Houghově prostoru se rovnice změní do podoby:

$$\mathbf{a} = x_i + \mathbf{r} \cdot cos(\theta_i) \tag{4.5}$$

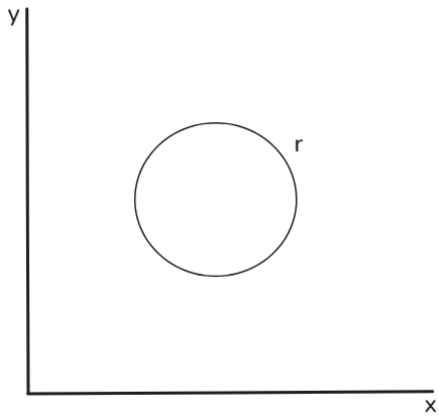
$$\mathbf{b} = y_i + \mathbf{r} \cdot sin(\theta_i) \tag{4.6}$$

Je zřejmé, že náš akumulátor nyní musí být trojrozměrný, na rozdíl od dvourozměrného, jako tomu bylo při detekci přímek[9]. Tím se jasně ukazuje souvislost mezi počtem hledaných parametrů a počtem dimenzí akumulátoru. Teoreticky můžeme přidávat další parametry a počet dimenzí poroste lineárně, avšak výpočetní náročnost roste exponenciálně a velmi rychle dosáhne bodu, kdy se Houghova transformace stává nepraktickou.

4.2.1 Princip

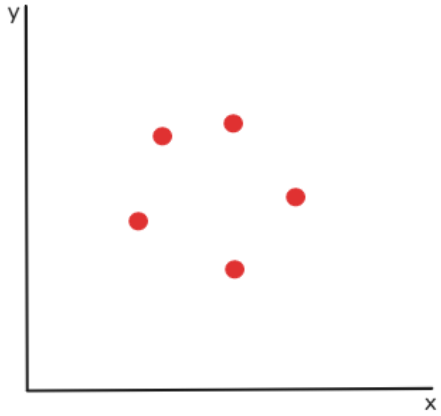
Stejným způsobem, jako jsme se snažili najít přímku v minulé sekci, se nyní pokusíme lokalizovat kruh, respektive kružnici.

Představme si, že se v obraze nachází objekt kruhovitého tvaru, např. se může jednat o mince, nádrže, míče nebo díry. Obrys takového objektu je zobrazen na obrázku 4.9.



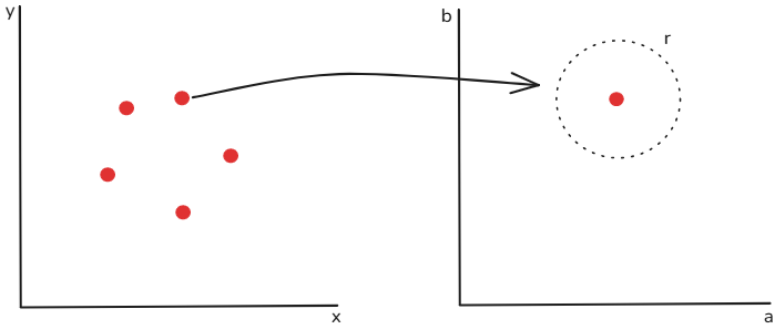
Obr. 4.9: Obrys hledaného objektu

Kružnice má určitý poloměr r. Předpokládejme, že předem známe jeho velikost. Stejně jako u přímek se kruh v obraze bude skládat z mnoha bodů ležících na jeho obvodu, jak je vidět na obr. 4.10.



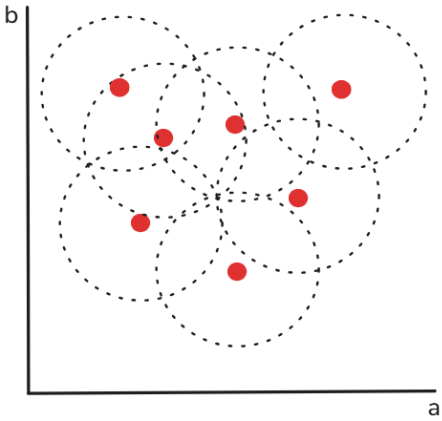
Obr. 4.10: Body reprezentující kružnici

Nyní se budou všechny body transformovat do Houghovy parametrické roviny podle rovnic 4.5 a 4.6. Tyto rovnice popisují kružnici, takže jejich parametrické obrazy budou mít tvar kružnic se středy v detekovaných bodech a stejným poloměrem r, jaký má i původní rovina obrazu.



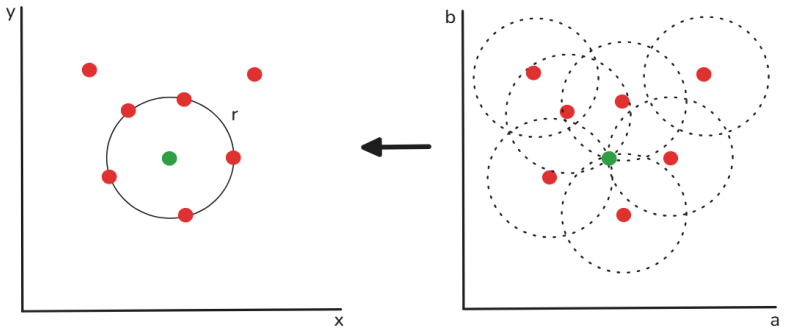
Obr. 4.11: Projekce bodů do kružnic

Projekce popsaná v předchozím odstavci je znázorněna na obr. 4.11. Je třeba si uvědomit, že tímto postupem vzniká kružnice kolem každého detekovaného místa, a tedy i kolem šumových struktur, které prošly předzpracováním obrazu (filtrováním, detekcí hran). Výsledkem je, že šum vytváří stejnou akumulační stopu jako skutečné body zájmu.



Obr. 4.12: Rovina parametrů s šumem

Na obr. 4.12 jsou zobrazeny všechny transformované body z obr. 4.10, doplněné o šum. Hlasování probíhá následovně: pro každý detekovaný bod se v akumulátoru zvýší hodnota v pozici a, b ve vzdálenosti r o jedna. Index, který obdrží dostatečný počet hlasů, je následně vyhodnocen jako lokalizovaný střed objektu – jak je patrné na obr. 4.13.

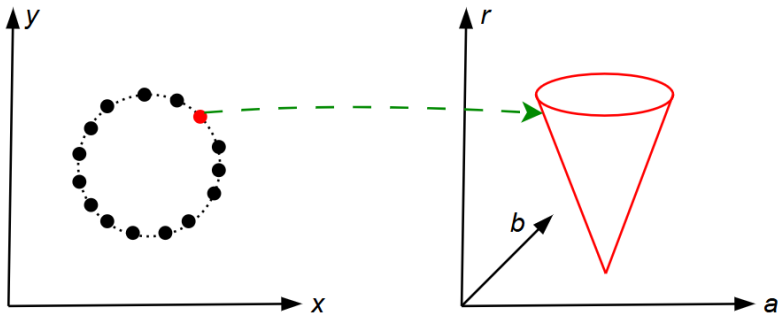


Obr. 4.13: Nalezená kružnice

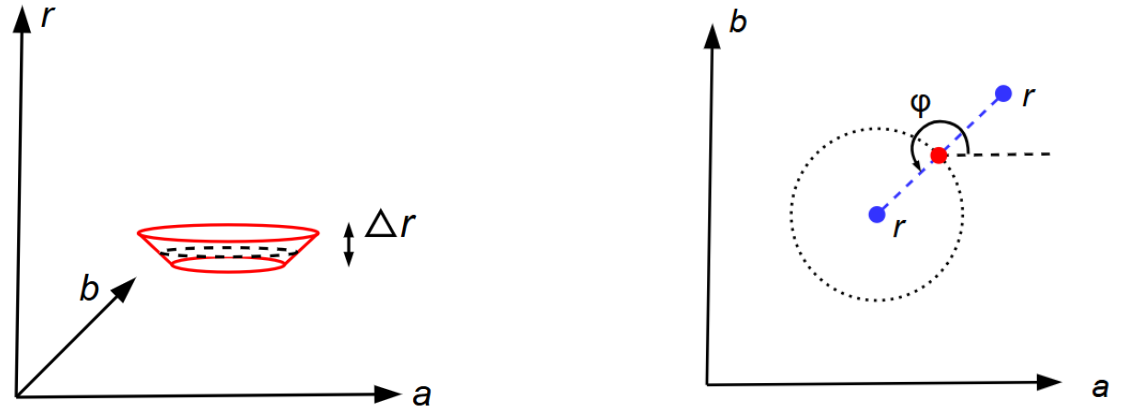
Určili jsme polohu hledaného objektu se známým poloměrem. V případě, že poloměr neznáme, zůstává postup obdobný – pouze roste výpočetní náročnost. Namísto inkrementace bodů tvořících kružnici ve 2D rovině budeme inkrementovat hodnoty ve 3D prostoru, kde se poloměr mění. Výsledný útvar v parametrickém prostoru pak bude povrch kužele. Znázorněno na obr. 4.14.

Pro zkrácení výpočtů a urychlení algoritmu se v praxi často využívá skutečnosti, že přibližný poloměr hledaného objektu bývá známý předem. Jedním ze způsobů je nastavení odchylky Δr v úzkém rozmezí okolo odhadované hodnoty, viz obr. 4.15a.

Alternativně lze kromě hran obrazu detekovat i jejich směr. U Houghovy transformace obecně platí, že čím více práce provedeme předem při zpracování obrazu, tím snazší je následná detekce. Takzvaná gradientní metoda umožňuje hlasovat pro menší počet bodů, čímž se výrazně snižuje výpočetní náročnost algoritmu.



Obr. 4.14: Transformace bodu v kužel



(a) Interval zkoumaných poloměrů

(b) Houghův gradient

Obr. 4.15: HT - metody hledání kružnic

Na obr. 4.15b je zobrazen princip Houghova gradientu, kdy se díváme na směr detekovaných pixelů. Jejich orientace je normála, kolmice na směr změny intenzity pixelu. Přestože sama normála má jasně daný úhel, pro jednoznačné určení středu kružnice nelze samotný směr použít. Nejjednodušší příklad je černý kruh na bílém pozadí vs bílý kruh na černém pozadí. Zatímco u bílého kruhu jdou šipky směrem dovnitř, u černého kruhu ukazují ven.

Z tohoto důvodu provádíme hlasování jak podél normály (úhel φ), tak v její opačné orientaci ($\varphi + 180°$).

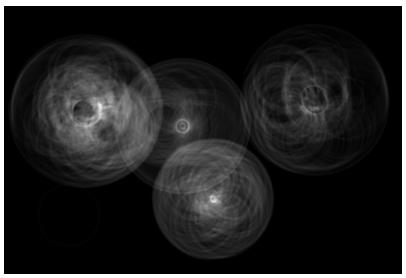
4.2.2 Ukázka detekce v Matlabu

Vstupní obraz (obr. 4.16) se skládá z mincí různých velikostí. Řekněme, že chci najít 1 Kč a dvě zlaté mince.

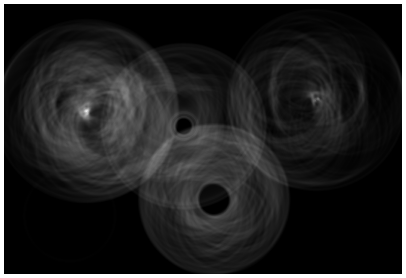
Jako první si obraz převedu do stupně šedi, vyfiltruji, aplikuji detekci hran a nakonec provedu Houghovu transformaci pro nalezení kružnic.

Obr. 4.16: Mince

Akumulátor pro 1 Kč je na obr. 4.17a a pro zlaté mince na obr. 4.17b.



(a)



(b)

Obr. 4.17: Akumulátor - mince

Detekované objekty jsou na obrázcích 4.18a a 4.18b.

(a)

(b)

Obr. 4.18: HT - nalezené mince

5 Obecná Houghova transformace

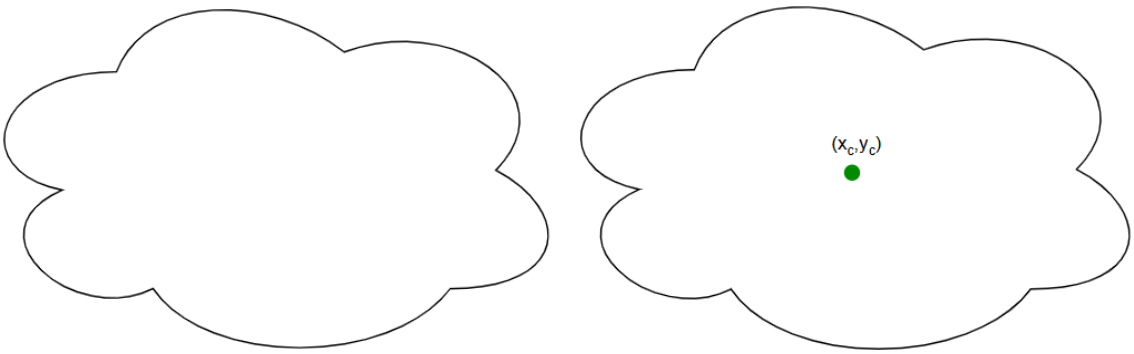
V předchozí části byla popsána klasická Houghova transformace a její schopnost detekovat základní geometrické útvary, jako jsou přímky nebo kružnice. V reálných situacích se ale často pracuje s tvary, které se nedají jednoduše vyjádřit rovnicí – například různé obrysy objektů nebo siluety. Právě pro takové případy vznikla obecná Houghova transformace (GHT), která rozšiřuje původní metodu a umožňuje vyhledávat i složitější nebo nepravidelné tvary.

GHT nepracuje s rovnicí tvaru, ale s předem připraveným modelem hledaného objektu. Ten je uložen ve formě tabulky, která ke každému směru hrany přiřazuje pozici vzhledem k referenčnímu bodu objektu. Při zpracování obrazu pak algoritmus sleduje, jak jednotlivé hrany odpovídají tomuto modelu, a ukládá výsledky do akumulačního prostoru. Tam, kde je shoda největší, se s největší pravděpodobností nachází hledaný tvar.

Vylepšený algoritmus HT představil profesor informatiky Dana H. Ballard v roce 1981.

5.1 Princip

GHT vyžaduje dva vstupní parametry. Prvním je binární obraz (logická mapa), označovaný jako *template*. Postup jeho vytvoření bude podrobně popsán v této kapitole. Druhým parametrem je obraz, ve kterém se snažíme daný *template* nalézt. Hlavní výhodou GHT je schopnost parametrizovat objekty, které nelze jednoduše vyjádřit analytickou rovnicí.



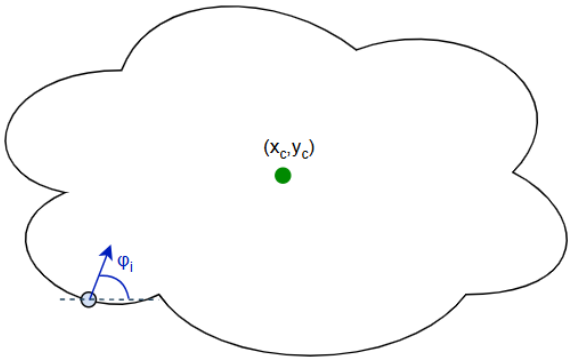
(a) Obrys mraku (b) Vyznačen referenční bod

Obr. 5.1: Arbitrální tvar (mrak) pro R-table

5.1.1 Tvorba R-table

Prvním krokem je volba referenčního bodu, ke kterému bude vztažena poloha a orientace všech bodů tvořících template. Nejčastěji se volí střed objektu, nicméně je možné zvolit libovolné místo v obraze. Pokud má například objekt těžiště výrazně posunuté na jednu stranu, je vhodné zvolit referenční bod blíže k těžišti, případně přímo v jeho místě.

Dále se zaměříme na jeden konkrétní bod hrany templatu, například ten zobrazený na obr. 5.2.

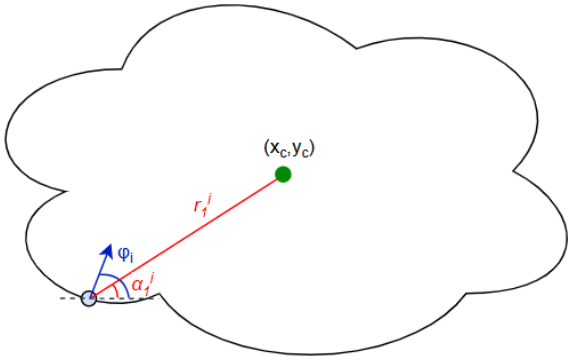


Obr. 5.2: Směr hrany

Po získání gradientu obrazu, tedy směru, kterým se mění intenzita pixelů, lze určit úhel φ, který daná hrana svírá s vodorovnou osou (0°). Tento úhel nabývá hodnot v rozsahu

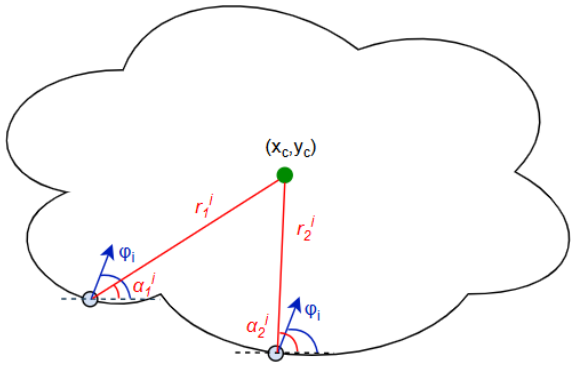
$$-\pi \leq \varphi \leq \pi$$

Následně definujeme vektor $\vec{r_k}$, který popisuje každý bod na hraně vzhledem k referenčnímu bodu. Tento vektor má dvě složky: první je vzdálenost r^i mezi bodem hrany a referenčním bodem, druhou je úhel α^i, který tento vektor svírá s vodorovnou osou (stejně jako φ).



Obr. 5.3: Mapování jednoho bodu

Důvodem, proč provádíme parametrizaci tímto způsobem, je skutečnost, že dvě nebo více oblastí v obraze mohou mít stejný gradient pixelu, a přesto se mohou nacházet ve zcela odlišných místech. Tato situace je ilustrována na obr. 5.4.



Obr. 5.4: Různé body - stejný směr

Všechny detekované hrany a jejich informace se uloží do jediné tabulky:

Tab. 5.1: R-table

Orientace hrany	$\vec{r} = (r,\alpha)$
φ_1	$\vec{r_1}^1, \vec{r_2}^1, \vec{r_3}^1$
φ_2	$\vec{r_1}^2, \vec{r_2}^2$
$\vdots$	$\vdots$
φ_n	$\vec{r_1}^n, \vec{r_2}^n, \vec{r_3}^n, \vec{r_4}^n$

Takto namapovaný útvar se nazývá Houghův model[10].

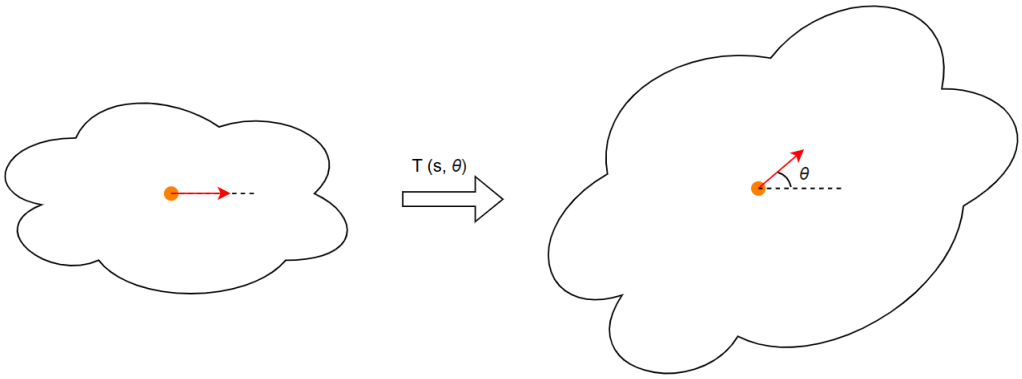
5.2 Možná vylepšení

Obecná Houghova transformace může být rozšířena o další parametry, které z ní činí velmi silný nástroj pro detekci tvarů. Nejčastějším rozšířením je úhel natočení objektu θ okolo referenčního bodu vzhledem k vodorovné ose (0°). Dalším běžně přidávaným parametrem je škálování (*scaling*). Funkce pro sestavení akumulátoru pak nabývá následující podoby:

$$\mathtt{H} = \{x, y, s, \theta\} \tag{5.1}$$

Kde souřadnice x a y určují 2D rovinu, zatímco škálování s a úhel natočení θ přidávají další dvě dimenze. Výsledný akumulátor pak náleží prostoru $\mathbb{R}^4$. Z toho je

patrné, že při práci s rozsáhlými daty se použití GHT v této podobě stává výpočetně velmi náročným a v praxi často nevhodným.



Obr. 5.5: Změna měřítka i rotace

5.3 Shrnutí

Houghova transformace se v digitálním zpracování obrazu hojně používá. Detekce přímek nachází své uplatnění v asistentech aut pro hlídání jízdního pruhu. Detekce kružnic například v kontrole kvality výrobků či při zkoumání CT snímků v medicíně. Obecná HT se zaměřuje spíše na rozpoznávání objektů[11].

Výhody HT:

- funguje na nespojených rozích
- relativně odolný vůči šumu a překrytí
- efektivní pro jednoduché tvary - HT
- pro složité tvary – GHT

Nevýhody GHT:

- velká paměťová náročnost
- velká výpočetní náročnost
- pro větší data prakticky nepoužitelné

6 Implementace algoritmu

Celý projekt se skládá z funkcí:

- `GenerateSinusoid`
- `Sin2Mat`
- `generateRTable`
- `computeAccumulator`
- `computeAccumulatorRotated`
- `findSuppressedMaxima`
- `findSuppressedMaxima3D`
- `estimateSinusoidParams`

První fáze algoritmu slouží k určení polohy sinusového signálu ve dvourozměrné rovině pomocí obecné Houghovy transformace. Po jeho lokalizaci následuje druhá fáze, jejímž cílem je získání parametrů daného signálu, konkrétně amplitudy a frekvence.

Po domluvě s vedoucím práce jsem upustil od *timeslice* obrazů a pokračoval ve zpracování vlastních snímků, které si sám generuji. Tyto obrazy se skládají ze dvou částí.

6.1 Funkce na generování signálu

První část tvoří funkce `GenerateSinusoid`, která vytváří jednorozměrný signál. Nejprve se navzorkuje časový vektor `t`, jenž se následně využije k výpočtu sinusového průběhu podle rovnice:

$$S(t) = DC + A \cdot sin(2\pi ft + \varphi) \cdot e^{-\alpha t} \tag{6.1}$$

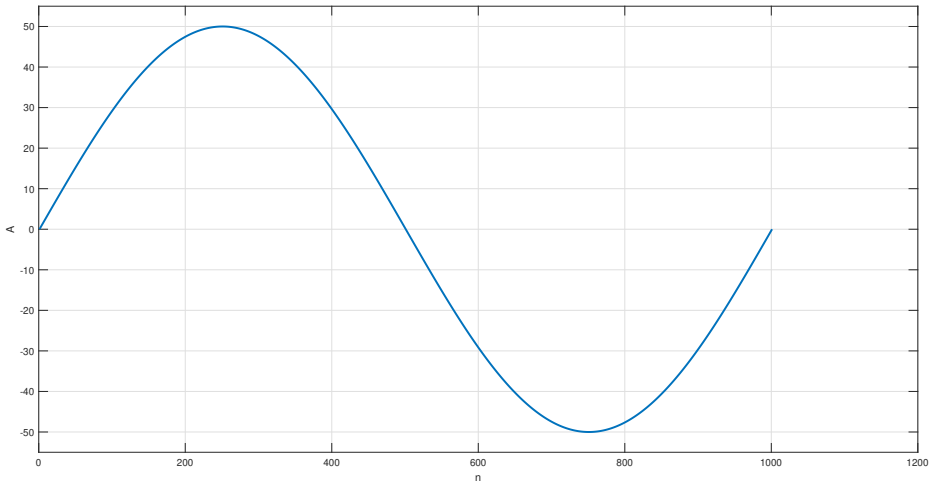
```
function [signal] = GenerateSinusoid(A,freq,time,Ts,alfa,DC, phase)
    t = 0:Ts:time;
    signal = DC + A * exp(-(alfa/100)*t) .* ...
             sin(freq*2*pi*t + deg2rad(phase));
end
```

Výpis 6.1: Funkce pro generování sinusoidy

Průběh vytvořeného signálu může vypadat např. jako na obr. 6.1.

6.2 Funkce na generování obrazu

Obraz ve zpracování vícerozměrných signálů představuje dvourozměrné pole hodnot - matici. V případě barevných snímků (např. RGB) se jedná o vícerozměrné



Obr. 6.1: 1D signál - funkce GenerateSinusoid

pole s dalšími kanály, nicméně základní zpracování stále vychází z práce s maticemi. Funkce `Sin2Mat` převádí jednorozměrný signál na binární matici, kde nula reprezentuje prázdné místo a jednička pozici, ve které se nachází vzorek signálu.

V původním návrhu této funkce byl výstup normalizován vzhledem k rozměrům obrazu. Takže amplituda předaného signálu neměla žádný vliv na velikost amplitudy v obraze. To se následně opravilo, jinak by nebylo možné jak určit daný parametr. Vstupními argumenty jsou:

`signal`	vektor z funkce `GenerateSinusoid`
`img_height`	výška obrazu
`img_width`	šířka obrazu
`method,L,Deg`	umělé rozšíření
`draw_method`	doplnění prázdných míst mezi vzorky

```
function [img] = Sin2Mat(signal, img_height, img_width, method, L, Deg,
    draw_method)
    arguments
        signal (:,:) double
        img_height (1,1) double {mustBePositive, mustBeNonzero, mustBeIntegerValue}
        img_width (1,1) double {mustBePositive, mustBeNonzero, mustBeIntegerValue}
        method (1,1) string {mustBeMember(method, ["disk", "line"])}
        L (1,1) double {mustBePositive, mustBeIntegerValue, mustBeNonzero}
        Deg (1,1) double {mustBeFinite}
        draw_method (1,1) string {mustBeMember(draw_method, ["bresenham",
            "morph"])} = "bresenham"
    end
    % Initialize output
    img = zeros(img_height, img_width,"single");

    % Center signal
    centered_signal = round(img_height/2 - signal);

    % Distribute points horizontally
    x_positions = round(linspace(1, img_width, length(signal)));

    % Draw the signal
    for i = 1:length(signal)-1
        % X-axis and Y-axis
        x1 = x_positions(i);
        x2 = x_positions(i+1);
        % Stay within image boundaries
        y1 = min(img_height, max(1, centered_signal(i)));
        y2 = min(img_height, max(1, centered_signal(i+1)));

        if strcmp(draw_method, "bresenham")
            points = Bresenham(x1, y1, x2, y2);
            for k = 1:size(points, 1)
                if points(k,1) >= 1 && points(k,1) <= img_width && ...
                   points(k,2) >= 1 && points(k,2) <= img_height
                    img(points(k,2), points(k,1)) = 1;
                end
            end
        else
            if x1 >= 1 && x1 <= img_width && y1 >= 1 && y1 <= img_height
                img(y1, x1) = 1;
            end
        end
    end

    % Apply morphological operation if needed
    if strcmp(method, "disk")
        img = imdilate(img, strel("disk", L));
    elseif strcmp(method, "line")
        img = imdilate(img, strel("line", L, Deg));
    end
end
```

Výpis 6.2: Funkce pro generování obrazu

Funkce `Sin2Mat` využívá vnořenou funkci `Bresenham` pro spojování prázdných míst mezi jednotlivými vzorky. Když bude vzorkování moc hrubé, povede to k ne-

spojité sinusoidě ve snímku. Důsledkem je málo bodů k detekci, což snižuje šanci na nalezení.

Bresenhamův algoritmus je klasický algoritmus pro vykreslování úseček v rastrové grafice. Určuje, které body na rastru je potřeba zvolit, aby co nejlépe aproximovaly přímku mezi dvěma zadanými body. Algoritmus využívá pouze celočíselné operace jako sčítání, odčítání a posuvy bitů, což zajišťuje jeho vysokou výpočetní efektivitu na běžných počítačových architekturách[12].

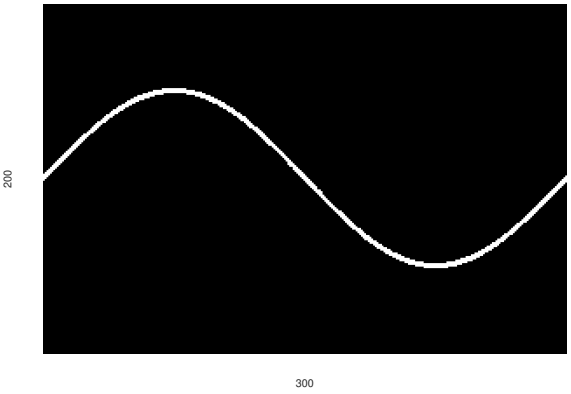
```
function points = Bresenham(x1, y1, x2, y2)
    % Ensure inputs are integers
    x1 = round(x1); y1 = round(y1);
    x2 = round(x2); y2 = round(y2);

    % Calculate differences
    dx = abs(x2 - x1);
    dy = abs(y2 - y1);

    % Get direction of next move
    sx = sign(x2 - x1);
    sy = sign(y2 - y1);
    err = dx - dy;
    points = [];

    while true
        points = [points; x1, y1];
        if x1 == x2 && y1 == y2
            break;
        end
        e2 = 2 * err;
        if e2 > -dy
            err = err - dy;
            x1 = x1 + sx;
        end
        if e2 < dx
            err = err + dx;
            y1 = y1 + sy;
        end
    end
end
```

Výpis 6.3: Bresenhamův algoritmus [16]

Funkci byl předán vstupní vektor z obr. 6.1 společně s požadovanými rozměry 200×300. Výsledný binární obraz je zobrazen na obr. 6.2. Je zřejmé, že amplituda 50 ze vstupního signálu vytvořeného funkcí `GenerateSinusoid` odpovídá amplitudě 50 pixelů ve výstupním obraze z funkce `Sin2Mat`.

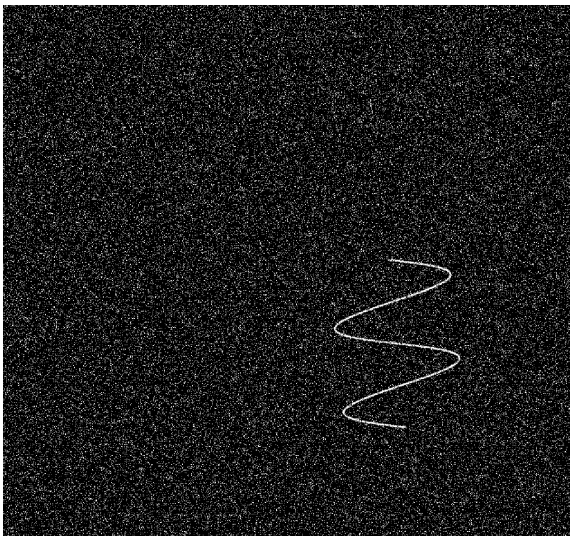
6.3 Příprava obrazu na detekci

Po vytvoření obrazu jsou jeho rozměry uměle zvětšeny doplněním nul, aby bylo možné následně provádět translace a rotace v libovolném rozsahu po celé ploše ob-



Obr. 6.2: 2D matice - funkce Sin2Mat

razu. Do takto připraveného snímku je následně přidán šum, nejčastěji ve formě "sůl a pepř" nebo šumu s gaussovským rozložením.



Obr. 6.3: Vstupní zašuměný obraz

6.3.1 Převod na stupně šedi

Pro detekování významných bodů se obvykle nepracuje přímo s barevnými snímky, ale převádějí se na šedotónový obraz.

Šedotónový obraz (anglicky grayscale image) je typ obrazové reprezentace, kde každý pixel nese pouze informaci o intenzitě světla, nikoli o barvě. Obraz se skládá výhradně z odstínů šedi, přičemž kontrast sahá od černé (nejnižší intenzita) po bílou

(nejvyšší intenzita). Takové obrazy jsou označovány jako černobílé nebo monochromatické a běžně se používají v mnoha oblastech zpracování obrazu, kde není potřeba barevná informace[13].

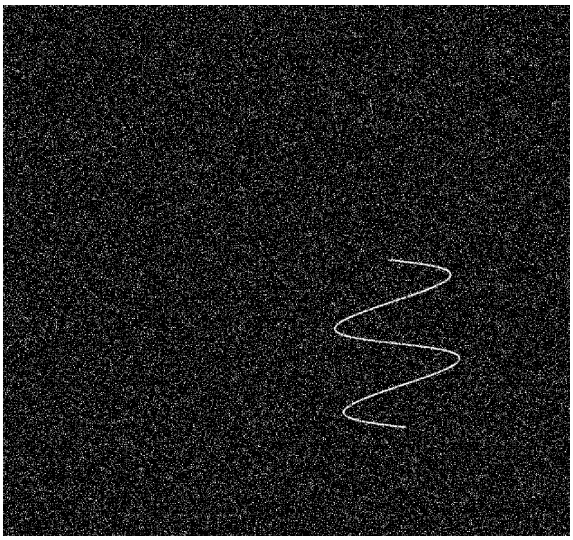
V Matlabu je již zabudovaná funkce `im2gray`, která používá model:

$$Y = 0.229R + 0.587G + 0.114B$$

využívající skutečnosti, že lidské oko je odlišně citlivé na jednotlivé složky RGB. Ve svém projektu jsem chtěl použít vlastní funkci `Image2gray`, která dává stejnou váhu každé složce:

$$Y = 1/3 \cdot R + 1/3 \cdot G + 1/3 \cdot B$$

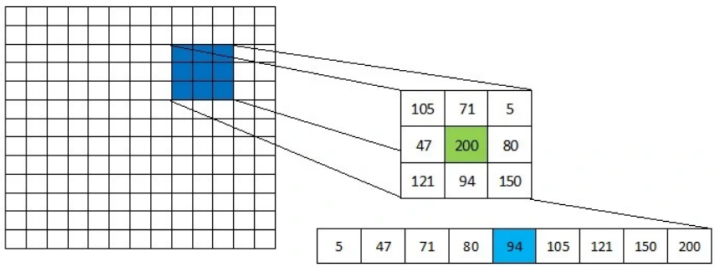
Ve výsledku použití funkce na převod do šedotónového snímku nemá vliv (obr. 6.4), protože vstupní matice je binární obraz.



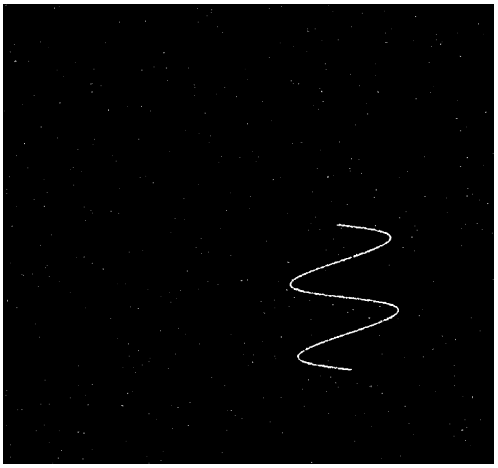
Obr. 6.4: Obraz ve stupních šedi

6.3.2 Filtrace

Pro účely filtrace existuje celá řada typů filtrů, jejichž volba závisí na charakteru očekávaného šumu ve zpracovávaném snímku. V tomto případě byl zvolen filtr založený na mediánovém principu, který je vhodný zejména pro potlačení šumu typu "sůl a pepř". Filtr pracuje tak, že v lokálním okně o rozměrech 3×3 se seřadí hodnoty jednotlivých pixelů vzestupně a výsledná hodnota středového pixelu je nahrazena mediánem z těchto hodnot.



Obr. 6.5: Princip median filtru [14]



Obr. 6.6: Obraz po filtraci

6.3.3 Detekce hran

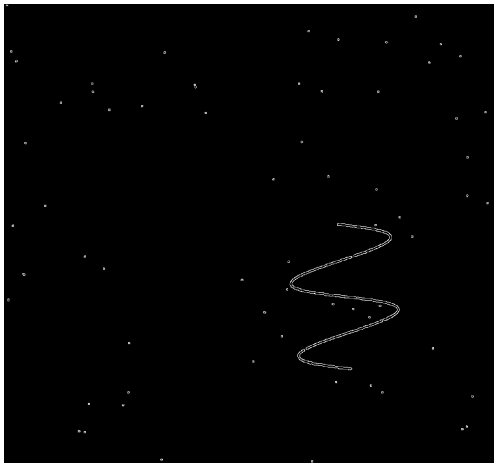
Ve své implementaci jsem zvolil Cannyho detektor hran. Jedná se o jeden z nejpoužívanějších algoritmů pro detekci hran v obraze, známý svou vysokou přesností a robustností vůči šumu. Sestává z několika na sebe navazujících kroků:

1. **Potlačení šumu** – Obraz je nejprve vyhlazen pomocí Gaussova filtru, čímž se sníží citlivost na šum.
2. **Výpočet gradientu** – Pomocí operátorů (např. Sobelových) se vypočítá velikost a směr intenzity gradientu, což ukazuje na pravděpodobné umístění hran.
3. **Potlačení ne-maxim** – V tomto kroku se zjemní linie hran tím, že se potlačí všechny pixely, které nejsou lokálním maximem ve směru gradientu.
4. **Prahování s hysterezí** – Aplikují se dva prahové limity: silné hrany (nad horním prahem) jsou vždy zachovány, zatímco slabé hrany (mezi prahy) jsou zachovány pouze tehdy, pokud navazují na silné hrany.

Výsledkem je binární obraz, ve kterém jsou výrazně označeny pouze ty hrany, které jsou relevantní a spojité. Díky kombinaci několika kroků je Cannyho detektor velmi

efektivní při zachování důležitých struktur a zároveň potlačuje falešné hrany způsobené šumem [15].

Po detekci hran je na výslednou matici aplikována morfologická operace *uzavření*. Tato operace umožní vyplnit sinusoidu, čímž se získá větší počet bodů vhodných k následné detekci. Vedlejší efekt však spočívá v tom, že dojde i k vyplnění zbylých shluků šumu, které prošly předchozí filtrací.



Obr. 6.7: Obraz po detekci hran

6.4 Detekce sinusoidy

Nyní máme připravený obraz, můžeme aplikovat první fázi - Obecná Houghova transformace. Nejdříve se vytvoří R-table z binární matice template. Pro návrh funkce `generateRTable` jsem se inspiroval algoritmem uvedeným v [17], který při sestavování tabulky využívá kartézské souřadnice namísto polárních, jak bylo popsáno v kapitole 5.1.1. Detekované hrany jsou mapovány na základě rozdílů vzdáleností v osách x a y vzhledem k referenčnímu bodu:

$$\begin{aligned} \Delta x &= x_i - x_0 \\ \Delta y &= y_i - y_0 \end{aligned}$$

Vstupy funkce:

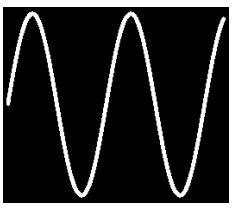
`temp`	matice hledaného tvaru
`rotationStep`	krok úhlu natočení
`MaxAngle`	maximální úhel natočení

Výstupy funkce:

`rtable`	namapovaná R-tabulka
`rtable_rot`	namapovaná R-tabulka s rotacemi
`binCount`	vektor uchovávající počet vektorů $\vec{r}$ na příslušný úhel

Postup funkce `generateRTable` lze shrnout následovně:

1. Nejprve jsou pomocí funkce `find` nalezeny souřadnice všech bodů, které tvoří hrany v předaném binárním obraze `temp`. Pokud nejsou nalezeny žádné hrany, funkce skončí s chybovým hlášením.
2. Následně je spočten směr hran pomocí funkce `imgradient`, která vrací úhly v rozsahu $[-180°, 180°]$. Ty jsou kvůli následnému indexování upraveny do rozsahu $[0°, 360°]$, protože MATLAB nepodporuje záporné indexy.
3. Vypočítá se počet rotačních kroků `numRotations` podle maximálního úhlu rotace `MaxAngle` a kroku `rotationStep`.
4. Jsou inicializovány výstupní struktury:
   - `rtable` – třírozměrné pole velikosti `[360 x maxP x 2]`, kde `maxP` je maximální počet bodů. Uchovává vektory z referenčního bodu ke každému bodu hrany, rozdělené podle směru.
   - `rtable\_rot` – čtyřrozměrné pole `[360 x maxP x 2 x numRotations]`, které obsahuje stejné vektory jako `rtable`, ale navíc i jejich rotace podle jednotlivých kroků.
   - `binCount` – vektor délky 360, který uchovává počet bodů spadajících do jednotlivých úhlových směrů.
5. Pro každý detekovaný bod hrany se provede:
   a) Určení příslušného směrového intervalu k dle úhlu gradientu.
   b) Výpočet vektoru od referenčního bodu ke zkoumanému bodu ve formátu $[\Delta x, \Delta y]$. Využívá se manhattanská vzdálenost.
   c) Uložení daného vektoru do odpovídajícího směrového intervalu v tabulce `rtable`.
   d) Dále se tento vektor rotuje pro všechny předem definované úhly a ukládá do čtvrté dimenze v `rtable_rot`.



Obr. 6.8: Template

6.4.1 R-table

```
function [rtable, rtable_rot, binCount] = generateRTable(temp, rotationStep,
    MaxAngle)
    % Extract edge pixel coordinates
    [x, y] = find(temp > 0);
    if isempty(x) || isempty(y)
        error("Template contains no edges.");
    end

    maxP = length(x);           % Maximum number of shape points
    numBins = 360;              % Use 360 bins for full 360deg resolution
    refPoint = round(size(temp) / 2);  % Center of template

    % Get edge directions
    [~, theta] = imgradient(temp);              % returns -180:180
    theta(theta < 0) = theta(theta < 0) + 360;  % shifts to 0:360

    % Rotation parameters
    numRotations = floor((MaxAngle - rotationStep) / rotationStep);

    % Initialize output
    rtable = zeros(numBins, maxP, 2, 'single');
    rtable_rot = zeros(numBins, maxP, 2, numRotations, 'single');
    binCount = zeros(numBins, 1, 'single');

    % For all edges
    for i = 1:maxP
        % Get angle bin
        k = round(theta(x(i), y(i)));
        if k > numBins, k = mod(k,numBins); end
        if k == 0, k = 1; end

        % Compute vector from reference point to current edge
        deltaX = x(i) - refPoint(1);
        deltaY = y(i) - refPoint(2);

        % Increment bin
        binCount(k) = binCount(k) + 1;
        h = binCount(k);

        % Write vector into R-table
        rtable(k, h, :) = [deltaX, deltaY];

        % Compute rotated vectors
        for n = 1:numRotations
            angle = deg2rad(n * rotationStep);
            rotX = round(cos(angle) * deltaX - sin(angle) * deltaY);
            rotY = round(sin(angle) * deltaX + cos(angle) * deltaY);
            rtable_rot(k, h, :, n) = [rotX; rotY];  % Store each rotation
                separately
        end
    end
end
```

Výpis 6.4: Funkce pro vytvoření R-table

6.4.2 Hlasování v akumulátoru

Pro návrh funkce `computeAccumulator` jsem se nechal inspirovat algoritmem uvedeným v [17]. Funkce provádí detekci objektu metodou GHT. Na základě hranového obrazu scény, směrové tabulky `rtable` a informací o počtu vektorů v jednotlivých směrových třídách (vektoru `binCount`) vyhodnocuje akumulační matici `ACC`, v níž vrcholy odpovídají možným pozicím středu hledaného objektu.

Průběh výpočtu sdílí s funkcí `generateRTable` kroky jako výpočet směru hran pomocí funkce `imgradient` a převod hodnot do rozsahu $[0°, 360°]$. Vlastní hlasování však probíhá na základě předem vytvořených směrových vektorů z R-tabulky.

U každého detekovaného bodu jsou zpětně promítnuty vektory z příslušného směrového intervalu (bin), čímž dochází k hlasování za možné pozice středu objektu v obraze. Hodnoty mimo obraz jsou filtrovány pomocí pomocné funkce `isValidPoint`.

Po ukončení hlasování je nalezena maximální hodnota v akumulační matici, která slouží jednak k orientačnímu vyhodnocení síly detekce, jednak jako rozhodovací prvek – pokud nedosáhne stanoveného prahu (250 hlasů), výstupní hodnota `maxACC` je nastavena na `NaN`, což signalizuje, že žádný výrazný kandidát na střed objektu nebyl nalezen.

Výsledná akumulační matice `ACC` je nakonec normalizována do rozsahu $[0, 1]$ a převedena na stejný datový typ jako tabulka `rtable`, což zajišťuje kompatibilitu s dalšími kroky zpracování.

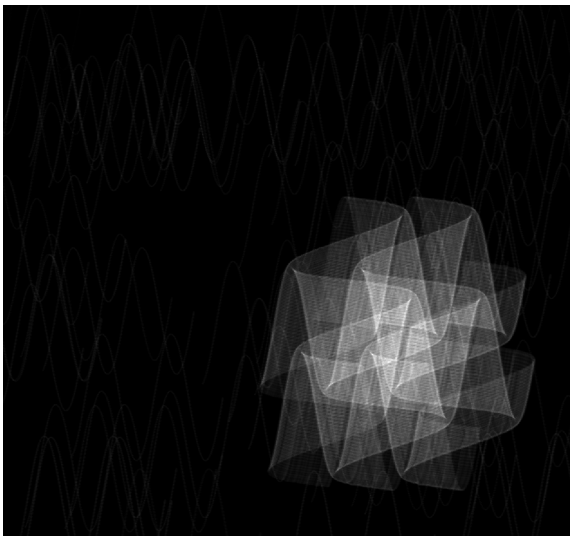
```
function [ACC, maxACC] = computeAccumulator(img, rtable, binCount)
    % Extract edge pixel coordinates
    [a, b] = find(img > 0);
    if isempty(a) || isempty(b)
        error("Accumulator is empty.");
    end

    % Get edge directions
    [~, theta] = imgradient(img);               % returns -180:180
    theta(theta < 0) = theta(theta < 0) + 360;  % shifts to 0:360

    % Initialize
    maxPoints = length(a);
    numBins = size(rtable, 1);
    ACC = zeros(size(img), 'single');

    % For all points
    for i = 1:maxPoints
        % Edge direction
        h = round(theta(a(i), b(i)));
        if h > numBins
            h = mod(h,numBins);
        end
        if h == 0, h = 1; end

        % For all vectors stored in binCount on h'th position
        for j = 1:binCount(h)
```

```
            dx = rtable(h, j, 1);
            dy = rtable(h, j, 2);
            % Calculate distance
            c = round(a(i) - dx);
            d = round(b(i) - dy);

            % Voting for points within image boundaries
            if isValidPoint(c, d, size(img))
                ACC(c, d) = ACC(c, d) + 1;
            end
        end
    end

    % For treshold setting
    maxACC = max(ACC(:));
    fprintf("Max in ACC is: %.0f\n", maxACC);

    % Determine whether shape was found or not
    if maxACC <= 250 % ~ 8 deg
        maxACC = NaN;
    end

    % Normalize to [0,1]
    ACC = cast(mat2gray(ACC),class(rtable));
end
```

Výpis 6.5: Funkce hlasování v akumulátoru

```
function valid = isValidPoint(c, d, size_img)
    valid = (c >= 1) && (c <= size_img(1)) && (d >= 1) && (d <= size_img(2));
end
```

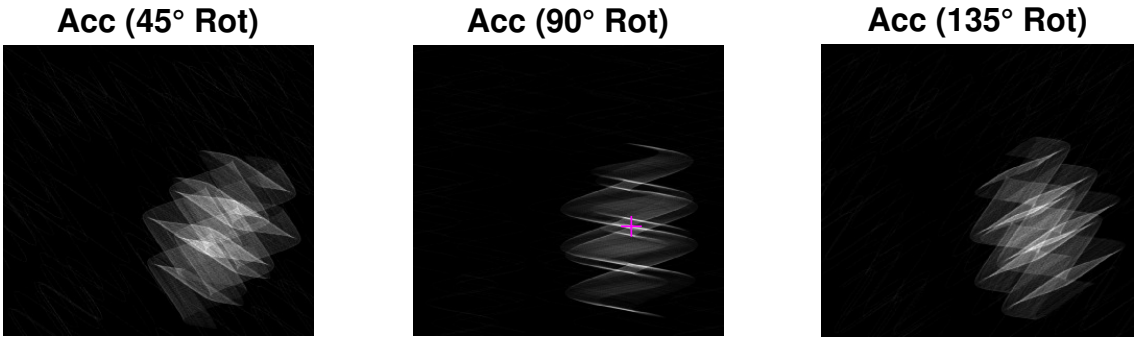
Výpis 6.6: Funkce pro kontrolu zapisování do rozměrů matice



Obr. 6.9: Akumulátor

Rotační varianta akumulátoru rozšiřuje základní verzi o schopnost detekovat objekt i při různých natočeních. Funkce `computeAccumulatorRotated` tedy využívá

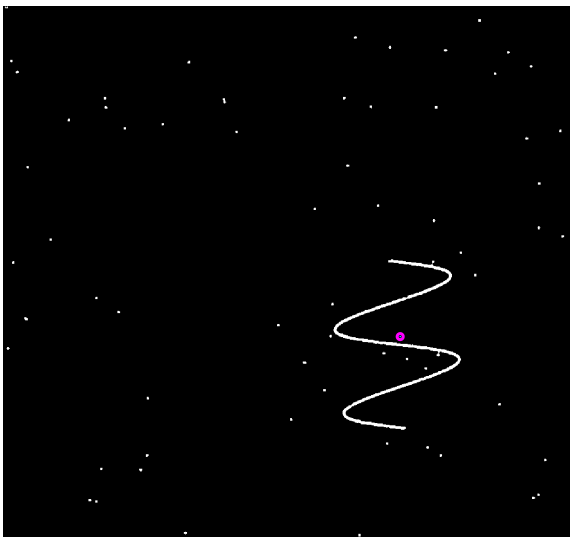
rozšířenou verzi směrové tabulky `rtable_rot`, která obsahuje vektory odpovídající různým rotačním variantám vzoru.



Obr. 6.10: Rotační akumulátor - jednotlivé rotace

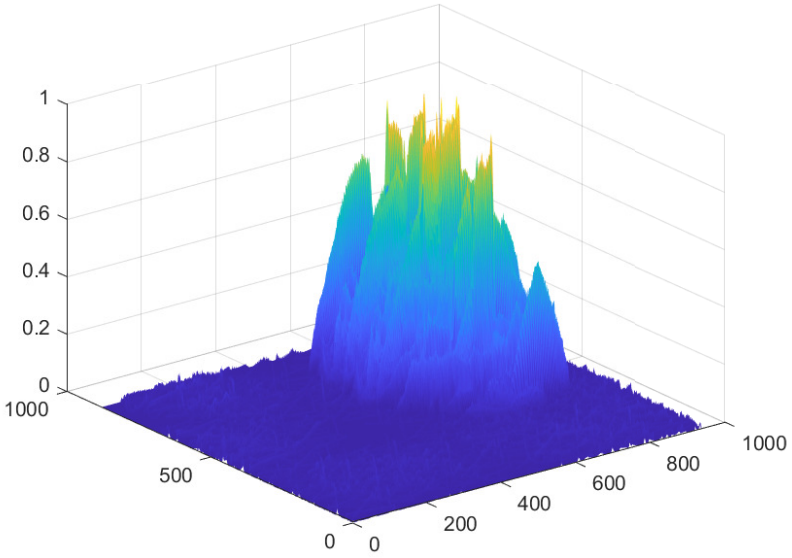
Princip hlasování zůstává v jádru stejný jako u klasického akumulátoru – pro každý bod s detekovanou hranou se podle jeho směrového intervalu zpětně promítají referenční vektory z R-tabulky. Hlasování však neprobíhá jen do jediné akumulační matice, ale do trojrozměrného pole `ACC_rotated`, kde třetí rozměr odpovídá různým úhlům natočení objektu.

Každý bod tak přispívá k více hypotézám – ke všem předem definovaným úhlovým variantám hledaného objektu. Výsledkem je množina akumulačních matic, z nichž lze identifikovat nejen pravděpodobnou polohu středu objektu, ale i jeho orientaci.



Obr. 6.11: Nalezená sinusoida

Tato metoda umožňuje detekci objektů nezávisle na jejich natočení v obraze, což je zásadní výhoda oproti základní verzi algoritmu.



Obr. 6.12: Rotační akumulátor - 3D

6.4.3 Suprese Maxima

Po výpočtu akumulačního prostoru je třeba z něj extrahovat nejvýznamnější body, které odpovídají pravděpodobné přítomnosti hledaného tvaru ve scéně. Samotná maxima v akumulátoru ale nestačí, protože v jeho okolí mohou existovat další podobně vysoké hodnoty způsobené šumem nebo malou odchylkou ve směru hran. Z toho důvodu se využívá tzv. potlačení netypických maxim (non-maximum suppression).

Tento krok lze chápat jako prostorovou filtraci, která vybírá pouze ta maxima, jež jsou nejvyšší v definovaném okolí (typicky čtvercovém). Postup zahrnuje několik kroků:

- Nejprve je akumulátor normalizován na interval [0,1], aby byl výběr podle prahové hodnoty konzistentní napříč různými scénami.
- Následně se odstraní všechna maxima, která nedosahují daného prahu (určeného například empiricky nebo relativně vůči celkovému maximu).
- Aby se předešlo situacím, kdy dvě shodné hodnoty v sousedství nejsou jednoznačně identifikovatelné jako maximum, dojde k jejich mírnému náhodnému rozlišení.
- Pomocí morfologické operace dilatace se nalezne, zda je daný bod skutečně nejvyšší v okolí daného poloměru (tedy lokální maximum).
- Výsledná maska kombinuje informace o prahování a lokálních maximech a slouží k určení souřadnic všech bodů, které splňují obě podmínky.

Výstupem jsou pak dvojice souřadnic (p,q), které reprezentují potenciální výskyty

hledaného tvaru v obraze.

```
function [p, q] = findSuppressedMaxima(ACC, threshold, radius)
    % Threshold the accumulator
    ACC = mat2gray(ACC);
    ACC(ACC < threshold) = 0;

    % Slightly randomize to break ties
    ACC = ACC - rand(size(ACC)) * 1e-6;

    % Create a binary mask of local maxima using dilation
    neighborhood = true(2 * radius + 1);  % square window
    localMax = imdilate(ACC, neighborhood) == ACC;

    % Combine thresholding and local max
    mask = localMax & (ACC > threshold);

    % Extract coordinates and values
    [p, q] = find(mask);
end
```

Výpis 6.7: Funkce pro zachování nejsilnějších hlasů

Pro nalezení výrazných lokálních maxim ve 3D akumulační matici je použit stejný princip potlačení ne-maxim rozšířený do tří rozměrů. Každý voxel akumulační matice je porovnán s hodnotami ve svém okolí definovaném 3D krychlí o zvoleném poloměru. Hodnoty, které nejsou lokálním maximem v tomto prostoru, jsou potlačeny. Před porovnáváním jsou navíc všechny hodnoty normalizovány a ty pod zvoleným prahem ignorovány. Výsledkem jsou souřadnice bodů, které představují významná maxima v celém 3D prostoru.

6.4.4 Estimace parametrů

Funkce má za úkol:

- vyříznout část obrazu v okolí daného bodu,
- natočit ji podle detekovaného směru vzoru,
- vypočítat frekvenci a amplitudu sinusového signálu obsaženého v tomto výřezu

1. **Zjištění velikostí:**
   - Zjistí rozměry vstupního obrazu a šablony (template).
   - Z templatu vypočte rozumnou velikost výřezu tak, aby pokryl oblast i při ne stoprocentní lokaci středu.
2. **Výběr souřadnic středu oblasti:**
   - Pokud není úhel (`angle`) definován nebo je maxACC zadán, použijí se souřadnice nerotovaného akumulátoru.
   - Jinak se použijí souřadnice rotovaného akumulátoru
3. **Bezpečný výřez obrazu:**

- Vyřízne okolí středu z původního obrazu, přičemž mimo hranice obrazu doplňuje nulami (zero-padding).

4. **Rotace oblasti:**
   - Pokud není zadáno maxACC, obraz se natočí opačným směrem, než je předaný úhel `angle`, aby byl co nejblíže vodorovné poloze.
5. Profil intenzit (součet přes řádky):
   - Vypočte se horizontální intenzitní profil (RowSum) – tedy součet pixelů v každém sloupci.
6. **Odhad frekvence:**
   - Najdou se lokální minima v tomto profilu pomocí `islocalmin`
   - Frekvence se odhaduje jako polovina počtu minim – tedy počet celých period.
7. **Odhad amplitudy:**
   - Vypočte se vertikální profil (`ColSum` – součet přes sloupce).
   - Vypočítá se adaptivní offset, který zohledňuje průměr a rozptyl.
   - Amplituda se pak odhaduje jako vzdálenost mezi první a poslední hodnotou, která přesáhla tento offset, dělená dvěma.

```
function [amp, freq, img_orig, img, profile,a,b,sidefile,offset] =
    estimateSinusoidParams(image, temp, x, y, x_rot, y_rot, angle, maxACC)

    % Get image and template sizes
    [imgH, imgW] = size(image);
    temp_size = round(size(temp)/2);
    tempH = ceil(temp_size(1) * 1.5);
    tempW = ceil(temp_size(2) * 1.5);

    if isnan(angle) || ~isnan(maxACC)
        x0 = x;
        y0 = y;
    else
        x0 = x_rot;
        y0 = y_rot;
    end

    % Safe cropping with zero-padding
    y1 = max(1, y0 - tempH);
    y2 = min(imgH, y0 + tempH);
    x1 = max(1, x0 - tempW);
    x2 = min(imgW, x0 + tempW);

    % Initialize with zeros
    img = zeros(2*tempH+1, 2*tempW+1, 'like', image);

    % Calculate destination coordinates
    dest_y1 = tempH + 1 - (y0 - y1);
    dest_y2 = tempH + 1 + (y2 - y0);
    dest_x1 = tempW + 1 - (x0 - x1);
    dest_x2 = tempW + 1 + (x2 - x0);

    % Place valid region
```

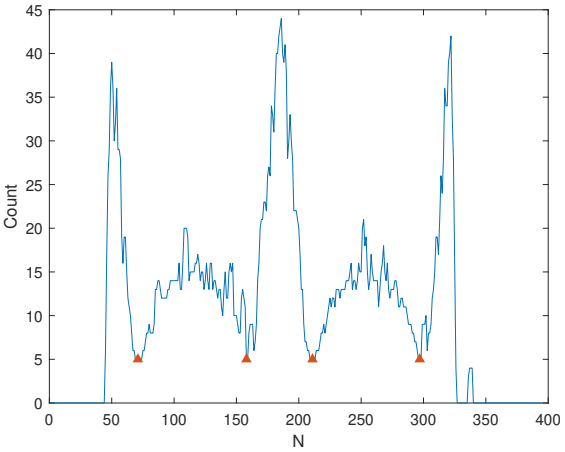
```
    img(dest_y1:dest_y2, dest_x1:dest_x2) = image(y1:y2, x1:x2);
    img_orig = img;

    if isnan(maxACC) % maxACC is stronger case than angle
        switch angle
            case 45
                img = imrotate(img,-angle,"bilinear","crop");
            case 90
                img = imrotate(img,-angle,"bilinear","crop");
            case 135
                img = imrotate(img,-angle,"bilinear","crop");
        end
    end

    profile = sum(img, 1);

    % Frequency estimation - localmin
    freq_idxs  =
        islocalmin(profile,"MinProminence",max(profile)/6,"ProminenceWindow", ...
                length(profile)/6,"MinSeparation",temp_size(2)/4);
    freqs = find(freq_idxs);
    if length(freqs) < 2
        error("Not enough frequency peaks in: estimateSinusoidParams");
    end
    a = find(freq_idxs);
    b = profile(freq_idxs);

    freq = length(freqs)/2;

    % Estimate Amplitude
    sidefile = sum(img, 2);

    offset = mean(sidefile) + 0.22 * std(sidefile); % adaptive offset

    amp = (find(sidefile > offset,1,"last") - find(sidefile > offset,1,"first"))
        /2;

    fprintf("Estimated amplitude: %.3f\n", amp);
    fprintf("Estimated frequency: %.3f\n", freq);
end
```

Výpis 6.8: Funkce pro estimaci parametrů ze signálu

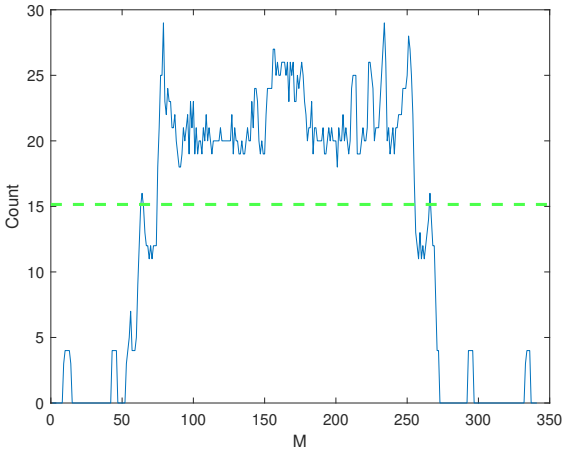
6.4.5 Získané parametry

Pro účely vyhodnocení byly použity synteticky generované snímky simulující časoprostorový průběh vibrací. Parametry sinusoidy (frekvence, amplituda, útlum) byly při generování známé, což umožnilo přímé srovnání s výsledky detekce. Do dat byl navíc uměle přidáván šum, aby bylo možné ověřit robustnost algoritmu i v méně ideálních podmínkách.

Tabulka 7.2 shrnuje jednak parametry šablony a jednak výsledky vyhodnocení z testovaného obrazu. Detekce frekvence je shodná s frekvencí signálu, podobně i odhad amplitudy odpovídá skutečnosti s malou odchylkou. Úhel natočení sinusoidy



Obr. 6.13: Sumace sloupců



Obr. 6.14: Sumace řádků

byl určen s větší nepřesností, to je přímý důsledek fixně nastavených úhlů rotace při výpočtu rotačního akumulátoru.

Tab. 6.1: Vyhodnocení sinusoidu

Template							
MxN	Freq	Amp	Decay				
226×262	2.2	105	0				
Image							
MxN	RealFreq	RealAmp	Decay	RealAngle	EstFreq	EstAmp	EstAngle
880×940	2	100	0	96	2	101	90

7 Výsledky studentské práce

Algoritmus byl navržen s ohledem na specifickou strukturu generovaných obrazů. Přestože byl testován za přítomnosti šumu, struktura snímků je stále velmi kontrolovaná – například pravidelnost tvaru a kontrastní pozadí jsou výrazně odlišné od toho, co by bylo možné očekávat u reálných záznamů. Při aplikaci na skutečné snímky s komplexnějším pozadím, neostrostí či nelinearitami by pravděpodobně bylo potřeba algoritmus upravit nebo přistoupit k odlišným metodám detekce.

Navržený postup tak zatím slouží spíše jako důkaz konceptu, že v řízeném prostředí je možné základní parametry vibrace určit. Pro rozšíření použitelnosti by bylo vhodné pokračovat v testování s realističtějšími daty.

Tab. 7.1: Vyhodnocení sinusoidy s útlumem

Template							
MxN	Freq	Amp	Decay				
226×262	1.8	108	20				
Image							
MxN	RealFreq	RealAmp	Decay	RealAngle	EstFreq	EstAmp	EstAngle
680×740	2	100	40	96	2	85.5	90

Tab. 7.2: Vyhodnocení sinusoidy

Template							
MxN	Freq	Amp	Decay				
270x315	2.7	75	0				
Image							
MxN	RealFreq	RealAmp	Decay	RealAngle	EstFreq	EstAmp	EstAngle
1000x1050	3	80	48	0	3	78	45

7.1 Limity práce

Tato sekce shrnuje hlavní problémy, na které algoritmus narážel, a navrhuje možná zlepšení pro budoucí práci.

7.1.1 Nepřesná lokalizace extrémů

Funkce `findSuppressedMaxima` je navržena tak, že potlačuje všechny lokální extrémy kromě jednoho nejvýznamnějšího. V případě více extrémů v blízkém okolí tak může dojít k posunu lokalizovaného středu, což vede k mírnému zkreslení detekovaných parametrů.

Možné řešení: V situacích, kdy je nalezeno více významných maxim, by bylo vhodné spočítat jejich vzdálenost a následně určit nový středový bod mezi nimi (např. aritmetickým průměrem jejich souřadnic). Tím by se zvýšila přesnost lokalizace, zejména u deformovaných nebo šumem ovlivněných obrazců.

7.1.2 Více lokálních minim při odhadu frekvence

Při numerické optimalizaci v rámci odhadu parametrů (zejména se jedná o frekvenci) může docházet ke vzniku více lokálních minim, což může vést k chybným výsledkům v závislosti na počátečním odhadu nebo intervalu hledání.

Možné řešení: Stejně jako u odhadu amplitudy lze použít heuristickou úpravu intervalu hledání. Konkrétně: omezit vyhledávání na interval definovaný jako průměr hodnot + standardní deviace a pracovat jen s tímto segmentem, čímž se sníží pravděpodobnost výběru nevhodného minima.

Závěr

Cílem této práce bylo navrhnout algoritmus pro odhad parametrů periodického signálu na základě obrazu reprezentujícího pohyb kmitajícího objektu. Vzhledem k tomu, že kód nebyl ve funkční fázi dostatečně brzo na zkoumání reálných obrazových dat, probíhalo testování výhradně na uměle vytvořených snímcích s přesně definovanými parametry.

Navržený algoritmus kombinuje detekci extrémů, lokalizaci oblasti zájmu a následnou optimalizaci pro určení frekvence, amplitudy a směru kmitání. Výsledky ukazují, že za předpokladu čistých nebo mírně zašuměných vstupů dokáže navržený postup dobře rekonstruovat parametry signálu. Složitější struktury nebo lokální chyby v lokalizaci ale mohou vést k určité nepřesnosti, zejména v případě, že dojde k potlačení relevantních maxim nebo k vícenásobnému výskytu lokálních minim.

Jednou z výhod je modularita navrženého řešení, které lze dále rozšiřovat a upravovat. V této souvislosti byl navržen i alternativní přístup založený na analýze směru gradientu v okolí extrémů sinusoidy. Tento postup byl částečně otestován na ideálně vodorovném průběhu pomocí skriptu `GradientFindingAmpFreq`, kde orientace gradientu ve špičkách sinusoidy (typicky kolem $\pm 90°$) sloužila jako vodítko pro určení frekvence a amplitudy. Při použití na obecně natočenou sinusoidu by však bylo nutné počítat s vychýlením směru a přizpůsobit způsob detekce směru gradientu. Tento přístup se jeví jako možná alternativa, zejména při zpracování reálných obrazových dat, kde by současný algoritmus narazil na své limity.

Literatura

[1] POR, Emiel; VAN KOOTEN, Maaike; SARKOVIC, Vanja *Nyquist–Shannon sampling theorem* [Online]. Leiden University, 2019. Dostupné z: https://home.strw.leidenuniv.nl/~por/AOT2019/docs/AOT_2019_Ex13_NyquistTheorem.pdf [cit. 2025-01-04]

[2] MURTHY, Akash. *Sampling Theorem - Digital Audio Fundamentals.* [Online]. Youtube.com. 2020. Dostupné z: https://www.youtube.com/watch?v=vrXGaFV1AmE&t. [cit. 2025-01-05].

[3] NAYAR, K. Shree. *Hough Transform | Boundary Detection.* [Online]. Youtube.com. 2021. Dostupné z: https://www.youtube.com/watch?v=XRBc_xkZREg. [cit. 2025-01-05].

[4] [Online]. Ascom-italy.it. 2022. Dostupné z: https://www.ascom-italy.it/product/gantry-cranes-industry/. [cit. 2025-01-06].

[5] Bracewell H. N.: *The Fourier Transform and its Applications.* 2nd ed. McGraw-Hill Book Company, New York 1986.[Online]. Dostupné z: https://archive.org/details/TheFourierTransformAndItsApplicationsBracewell/page/n25/mode/2up?view=theater. [cit. 2025-01-07].

[6] PRESS, William H. *Numerical recipes: the art of scientific computing.* 3rd ed. pg.602 Cambridge: Cambridge University Press, 2007. ISBN 978-0-521-88407-5. [Online]. Dostupné z: https://faculty.kfupm.edu.sa/phys/aanaqvi/Numerical%20Recipes-The%20Art%20of%20Scientific%20Computing%203rd%20Edition%20(Press%20et%20al).pdf [cit. 2025-01-07]

[7] NAYAR, K. Shree. *Fourier Transform | Image Processing II.* [Online]. Youtube.com. 2021. Dostupné z: https://www.youtube.com/watch?v=tEzgtbnbXgQ. [cit. 2025-01-07].

[8] Sofia Raffia Table Lamp. [Online]. Paolaandjoy. Dostupné z: https://paolaandjoy.com.au/products/sofia-raffia-table-lamp. [cit. 2025-01-07].

[9] PEDERSEN, Simon Just Kjeldgaard. *Circular Hough Transform* [online]. Aalborg University, Vision, Graphics, and Interactive Systems, 2007 Dostupné z: https://cdn.manesht.ir/9961___Simon_Pedersen_CircularHoughTransform.pdf [cit. 2025-05-17].

[10] NAYAR, K. Shree. *Generalized Hough Transform / Boundary Detection.* [Online]. Dostupné z: https://www.youtube.com/watch?v=_mGxmZWs9Zw. [cit. 2025-05-22].

[11] *Hough transform in computer vision.* [Online]. Dostupné z: https://www.geeksforgeeks.org/hough-transform-in-computer-vision/. [cit. 2025-05-24].

[12] *Bresenham's line algorithm.* In: Wikipedia: the free encyclopedia [Online]. Dostupné z: https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm#cite_note-Zingl-3. [cit. 2025-05-26].

[13] *Grayscale.* In: Wikipedia: the free encyclopedia [Online]. Dostupné z: https://en.wikipedia.org/wiki/Grayscale#cite_note-1. [cit. 2025-05-26].

[14] [Online]. Dostupné z: https://fiveko.com/wp-content/uploads/2017/08/median-filter-3x3-window.webp. [cit. 2025-05-26].

[15] GONZALEZ, Rafael C. a WOODS, Richard E. *Digital Image Processing 4th. Edition.* [Online]. Dostupné z: https://www.cl72.org/090imagePLib/books/Gonzales,Woods-Digital.Image.Processing.4th.Edition.pdf#page=732. [cit. 2025-05-27].

[16] ZINGL, Alois. [Online]. Dostupné z: https://zingl.github.io/Bresenham.pdf#page=13. [cit. 2025-05-26].

[17] SHELLY, Han. *Generalized-Hough-Transform.* [Online]. Dostupné z: https://github.com/Shellyhan/Generalized-Hough-Transform. [cit. 2025-05-24].