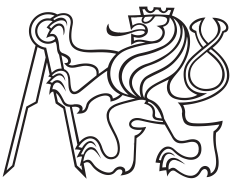
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Kamerový systém pro snímání vibrací

Jakub Vodseďálek
Kybernetika a robotika
Senzory a přístrojová technika

2016/2017
Vedoucí práce: doc. Ing. Jan Fischer, CSc.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

Akademický rok **2016-17**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	**Jakub Vodseďálek**
Studijní program:	**Kybernetika a robotika**
Obor:	**Senzory a přístrojová technika**
Název tématu česky:	**Kamerový systém pro snímání vibrací**
Název tématu anglicky:	**Camera Based Vibration Sensing System**

Pokyny pro vypracování:

Navrhněte a realizujte systém pro snímání vibrací, který bude pro určení okamžité polohy pohybujícího se objektu využívat obrazovou informaci z kamery CMOS. Obrazový senzor CMOS v kameře bude ovládán mikrořadičem řady STM32, který zajistí zachycení sekvence snímků v přesně definovaných okamžicích, které budou odvozeny od předpokládané periody vibrací tak, aby ze získané obrazové sekvence bylo možno vyhodnotit velikost pohybu objektu. Vytvořte potřebné programy pro mikrořadič i pro nadřazené PC. Prakticky ověřte možnost realizace takového systému a stanovte jeho možnosti a použitelnost pro zjištění nárůstu amplitudy vibrací objektů typu elektrický motor.

Seznam odborné literatury:

[1] Yiu, J.: The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors,
[2] STMicroelectronics: RM0090, STM32F4 Reference manual
[3] STMicroelectronics: DS9405- STM32F429 Data

Vedoucí bakalářské práce:	doc. Ing. Jan Fischer, CSc.
Datum zadání bakalářské práce:	24. ledna 2017
Platnost zadání do[1]:	30. září 2018

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická

Prof. Ing. Jan Holub, Ph.D.
vedoucí katedry

Prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 24. 1. 2017

[1] Platnost zadání je omezena na dobu tří následujících semestrů.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Tato práce vznikla v laboratoři videometrie, katedry měření ČVUT - FEL v Praze pod vedením doc. Ing. Jana Fischera, CSc. Navazuje též na výzkum v rámci MSM6840770015 - „Výzkum metod a systémů pro měření fyzikálních veličin a zpracování naměřených dat“ a další práce řešené v laboratoři videometrie, jejichž některé poznatky a výstupy v oblasti aplikace optoelektronických senzorů také využívá.

V Praze dne 24. 5. 2017 .....

Poděkování

Rád bych zde poděkoval za veškerou podporu, které se mi při tvoření této bakalářské práce a během celého studia dostalo. Hlavní poděkování patří vedoucímu této práce doc. Ing. Janu Fischerovi, CSc. za neocenitelné rady a připomínky při řešení problémů nejen s bakalářskou prací. Dále bych chtěl poděkovat doc. Ing. Radislavu Šmídovi, Ph.D. za možnost tuto práci vytvářet a také za přínosné informace. V neposlední řadě patří velké díky mé rodině a přátelům za podporu a důvěru během celého studia.

Abstrakt

Tato práce se zabývá návrhem a realizací systému pro snímání vibrací s využitím obrazové informace z CMOS senzoru. CMOS senzor je řízen mikrořadičem z řady STM32, který má za úkol řízení odběru snímků v přesně definované okamžiky odvozené z předpokládané frekvence vibrací. Použitím metod zpracování obrazu je následně z obrazové informace získán průběh polohy měřeného objektu. Výsledkem je potom informace o amplitudě měřených vibrací. V rámci práce jsou vytvořeny programy pro mikrořadič a nadřazené PC. Cílem je především ověření možností detekce vibrací pomocí informace z obrazového CMOS senzoru.

Klíčová slova: STM32F429, CMOS obrazový senzor, detekce vibrací, virtual COM port, vibrace, stroboskopické snímkování, zpracování obrazu, hranová detekce

Abstract

This bachelor thesis deals with the design and implementation of a vibration sensing system using image information from the CMOS sensor. The CMOS sensor is controlled by the STM32 series microcontroller, which has the task of controlling the sampling at precisely defined moments derived from the expected vibration frequency. Using image processing methods, the position waveform of the measured object is then acquired from the image information. The goal is then to compute the amplitude of the measured vibrations. The programs for microcontroller and parent PC are created within the work. The main objective is to verify the possibilities of detecting vibrations using information from a CMOS image sensor.

Keywords: STM32F429, CMOS image sensor, vibration sensing, virtual COM port, vibration, stroboscopic capture, image processing, edge detection

Title translation: Camera Based Vibration Sensing System

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Pro detekci vibrací lze v dnešní době použít hned několik možných způsobů. Cílem této práce je ověření možností detekce vibrací pomocí informace z obrazového CMOS senzoru. Tato možnost je jistě jedna z těch méně běžných. Obecně však optická měření stále více získávají na oblibě. Se zvyšujícím se výkonem vestavěných systému se v poslední době začínají také rozšiřovat systémy založené na informaci právě z obrazových senzorů.

Možnou aplikací je potom především diagnostika vibrací průmyslových strojů. V těchto případech jsou většinou používány dotykové senzory vibrací. Systém využívající informace z obrazového senzoru by jistě mohl být bezkontaktní, což by byla jeho nesporná výhoda. Takový systém by totiž mohl snímat i vibrace rotačních částí strojů nebo jiných částí nějakým způsobem nevhodných na umístění kontaktního senzoru. Jedna sestava by také při vhodné konfiguraci mohla snímat hned několik částí stroje najednou.

Hlavním problémem je nutnost násobně vyšší snímkovací frekvence senzoru pro bezpečné určení amplitudy snímaných vibrací. Kamery dosahující dostatečných parametrů jsou cenově nedostupné. Řešení se pokusíme najít ve stroboskopickém snímkování. Při jeho použití se spokojíme s použitím CMOS obrazového senzoru v dostupné cenové relaci. Přesné časování odběru snímku bude zajištěno řízením senzoru pomocí mikrořadiče.

Cílem práce je především ověření schopnosti detekce vibrací na subpixelové úrovni pomocí CMOS obrazového senzoru řízeného mikrořadičem. V rámci práce bude vytvořen program pro mikrořadič a testovací aplikace pro nadřazený počítač. Protože jde především o ověření možností takovéto detekce, bude veškeré zpracování obrazových dat probíhat v nadřazeném počítači. Tímto způsobem je možné vyzkoušet větší množství přístupů a zároveň lépe kontrolovat výsledky.

Kapitola 2
Analýza problematiky

Tato práce si klade za cíl ověření možností detekce vibrací pomocí informace z obrazového CMOS senzoru. Detekce vibrací bude převedena na měření změny polohy snímaného objektu. Abychom byli schopni bezpečně určit amplitudu výchylky, musí být snímací frekvence výrazně vyšší nežli frekvence snímaných vibrací. Pro splnění tohoto kritéria by ale byla potřeba ne velmi dobře cenově dostupná kamera.

Protože jsou ale vibrace periodický pohyb, je možné použít stroboskopického snímkování (snímkování v ekvivalentním čase). Jeho použití klade velmi přísné požadavky na časování odběru snímku. Této přesnosti je možné dosáhnou díky řízení obrazového senzoru pomocí mikrořadiče.

Při snímání jakéhokoli pohybu je důležité co nejvíce zamezit pohybovému rozostření. To je možné pomocí snižování doby expozice. Při příliš krátké době expozice však senzor nestihne nabrat dostatek světla a snímek je nepoužitelný. Nejjednodušším řešením tohoto problému je použití přisvětlení. Tím zajistíme dostatek světla při krátké době expozice a zároveň i zmírníme vliv změny okolního osvětlení.

V této kapitole jsou dále obecně rozebrány možnosti měření vibrací. Větší důraz je kladen na možnosti obrazových senzorů, které jsou hlavním prvkem této práce. Nakonec jsou nastíněna možná řešení realizace celé sestavy.

2.1 Senzory vibrací

Měření vibrací je ve většině případů převáděno buď na měření zrychlení nebo na měření změny polohy objektu. V této kapitole budeme z větší části vycházet z [1]. Všechny senzory vibrací můžeme obecně rozdělit do tří skupin:

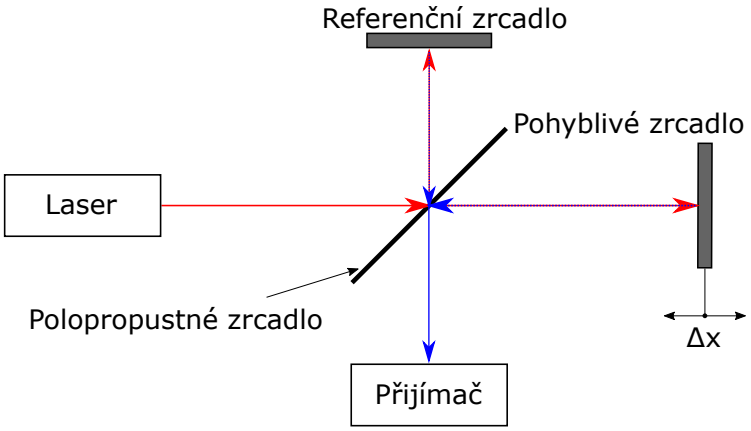
- Mechanické
- Interferenční
- Bezkontaktní

2.1.1 Mechanické senzory vibrací

U tohoto druhu senzorů se ve většině případu převádí měření vibrací na měření zrychlení pomocí akcelerometrů. Výstupem takového senzoru je potom hodnota přímo úměrná zrychlení v měřeném směru. Nevýhodou těchto senzorů je, že musí být vždy v kontaktu s měřeným objektem. Používají se tedy na měření nerotačních součástí strojů, kde je možné takové senzory instalovat. Výhodou je potom jednoduchost a především cena senzoru.

2.1.2 Interferenční senzory vibrací

Jak již název napovídá tyto senzory fungují na principu interference. Interference je jev, kdy dochází ke skládání dvou nebo více vlnění o stejné frekvenci. Při fázovém posunutí mezi vlněními potom dochází buď ke konstruktivní nebo destruktivní interferenci. K plně



Obrázek 2.1. Blokové schéma interferenčního senzoru polohy.

konstruktivní interferenci dojde, když jsou vlnění ve fázi. Na druhou stranu plně destruktivní interference nastane pokud jsou vlnění vzájemně posunuta o půl periody (v opačné fázi).

Jako zdroj měřeného záření se v praxi používá laser. Ten svítí skrze polopropustné zrcadlo na referenční a pohyblivé zrcadlo umístěné na měřeném objektu. Od těchto zrcadel se paprsek odráží směrem k přijímači, kde je vyhodnocena interference dopadajících vlnění. Jak se mění vzdálenost pohyblivého zrcadla, mění se i fázový posuv mezi vlněním odraženým od něj a od statického zrcadla. Interferenční senzory vibrací tedy převádí měření vibrací na měření velikosti výchylky ve směru vibrací. Základní schéma takového senzoru můžeme vidět na obrázku 2.1. Hlavní přednostní laserové interference je její přesnost, která se může pohybovat až okolo 1 nm. Za nevýhody můžeme počítat vyšší cenu, složitou instalaci a nutnost umístění části sestavy na měřený prvek. Je také nutné znát předem směr měřeného pohybu (měří pouze v tečném směru k paprsku laseru).

2.1.3 Bezkontaktní senzory vibrací

Na rozdíl od mechanických senzorů vibrací měří bezkontaktní senzory vibrací výchylku polohy. Převádějí tedy stejně jako interferenční senzory měření vibrací na měření výchylky (změny polohy) ve směru vibrací. Senzorů spadajících do této kategorie je ovšem velké množství a jejich popsání by vystačilo na celou bakalářskou práci. Tato práce se zaměřuje pouze na jeden z možných způsobů měření a ostatní senzory budou tedy pouze zmíněny. Každý z těchto typů senzorů má hned několik možností realizace a všechny mají své výhody i nevýhody. Mezi základní typy bezkontaktních senzorů vibrací patří:

- Indukční
- Indukčnostní
- Kapacitní
- Optické

Zabývat se zde budeme relativně mladou možností detekce vibrací (pohybu). Z opatrností bychom ji možná mohli zařadit do kategorie optických senzorů. Od funkce senzorů z této kategorie se ale stále zásadně liší. Jedná se totiž o detekci vibrací v obrazové sekvenci. Tato metoda využívá algoritmů pro zpracování obrazu a ze sekvence snímků se snaží získat informaci o směru a velikosti pohybu. Tento postup je silně závislý na použitých algoritmech a vyhodnocování dat je výrazně složitější než u jiných postupů. Zároveň ale tento přístup umožňuje širokou škálu aplikací a je velmi univerzální.

2.2 Obrazové senzory

Obrazové maticové senzory můžeme rozdělit dle technologie do dvou kategorií. Jedna z technologií je známa pod zkratkou CCD (Charge-coupled device) a druhá pod zkratkou CMOS (Complementary Metal–Oxide–Semiconductor). Předmětem této práce není do hloubky rozebírat principy a vlastnosti obou technologií, ale je vhodné alespoň zmínit hlavní rozdíly. V celé kapitole je vycházeno především z [2].

2.2.1 CCD obrazové senzory

CCD obrazové senzory jsou vývojově starší. I přes to mají stále několik nepřekonaných výhod a proto jsou stále používány. Především jsou díky zesilování signálů po řádcích méně citlivé na šum. Mají také vyšší světelnou citlivost. Na druhou stranu mají již od začátku i neduhy. Při přílišném zasvícení na senzor dochází k efektu nazývanému smear (stěr), při kterém se obraz „rozteče“ a vytvoří světlé pruhy. Od začátku je také výroba těchto senzorů nákladná. CCD obrazové senzory jsou především používány v dražší technice, kde je dbáno na maximální kvalitu obrazu a nehledí se na cenu.

2.2.2 CMOS obrazové senzory

CMOS obrazové senzory využívají výrobních technologií podobných těm u integrovaných obvodů a procesorů. Jejich výroba je velmi levná, což je do začátku jedna z hlavních výhod oproti CCD obrazovým senzorům. K dalším výhodám potom patří až několikanásobně nižší spotřeba a vyšší pracovní rychlost. Hlavní nevýhodou je potom vyšší citlivost na šum, způsobená zesilovačem u každé světlocitlivé buňky. CMOS obrazové senzory stále získávají na oblibě především díky své příznivé ceně. Obrazovou kvalitou už se také téměř dotahují na CCD obrazové senzory a je tedy možné, že v budoucnu budou CCD obrazové senzory téměř vytlačeny.

	CCD	CMOS
Cena	-	+
Citlivost na šum	+	-
Spotřeba	-	+
Kvalita obrazu	+	-
Rychlost	-	+

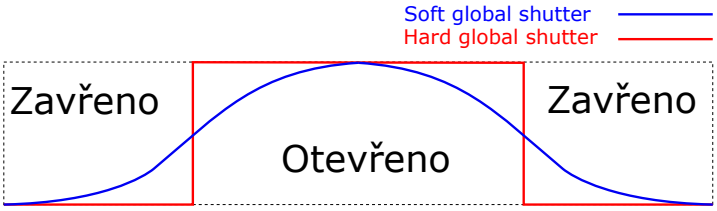
Tabulka 2.1. Porovnání CCD a CMOS obrazových senzorů.

2.2.3 Řízení závěrky

Obrazové maticové senzory můžeme dále rozdělit podle způsobu ovládání závěrky (shutter). Obě technologie senzorů (CCD i CMOS) obecně podporují jak global tak rolling shutter. Srovnání těchto dvou způsobů je pro tuto práci důležitější než samotné technologie senzorů. Oba přináší nové možnosti pro realizaci měřící soustavy a bude jim tedy věnováno více prostoru.

2.2.4 Global shutter

Při použití global shutter dochází k expozici všech světlocitlivých buněk senzoru v jeden okamžik. Závěrka je tedy vždy pro celý senzor ve stejném stavu (Open/Close). Tento způsob expozice je na první pohled vhodný pro snímání rychle se pohybujících objektů. Díky expozici všech světlocitlivých buněk najednou, může dojít k rozmazání obrazu pouze při použití příliš dlouhé doby expozice. Existují dva přístupy k použití tohoto typu expozice,



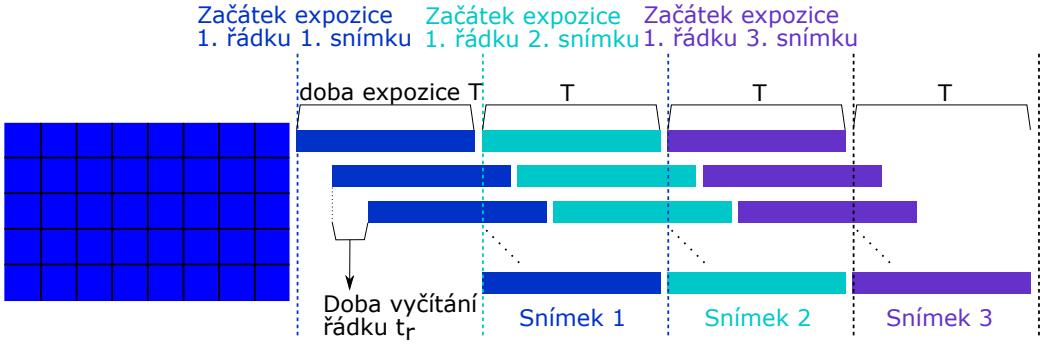
Obrázek 2.2. Porovnání soft a hard global shutteru.

které jsou někdy označovány jako hard a soft shutter [3]. Rozdíl mezi nimi nejlépe ilustruje obrázek 2.2.

Zejména u CMOS senzorů přináší global shutter i jisté nevýhody. Vzhledem k ukončení expozice celého snímku v jeden okamžik, je nutné získané náboje reprezentující jednotlivé pixely přenést a uschovat. A/D převodník následně provádí postupnou digitalizaci nasbíraných dat. Před započtením další expozice je také nutné provést globální vyčištění senzoru od nábojů naakumulovaných během vyčítání. Kvůli tomuto nutnému postupu je efektivní rychlost oproti rolling shutter módu snížena až na polovinu. Delší prodleva mezi nasbíráním a digitalizací dat také zvyšuje množství šumu.

2.2.5 Rolling shutter

Vyšší rychlosti senzoru je při použití rolling shutteru dosahováno pomocí postupné expozice světlocitlivých buněk senzoru. Před dokončením expozice jednoho snímku je možné začít expozici snímku dalšího. Tento proces je ve většině případů natolik rychlý, že je ve výsledném obraze nepostřehnutelný. Čas potřebný k vyčtení řádku ovšem nelze eliminovat a projevuje se jako zpoždění mezi expozicemi jednotlivých řádků. Toto zpoždění způsobuje překrývání snímků 2.3 a může zapříčinit zkreslení obrazu při sledování rychle se pohybujícího objektu. Toto zkreslení také někdy nazýváme zkreslením prostorovým.



Obrázek 2.3. Princip překrývání snímků u rolling shutteru.

Pro spočtení doby expozice, pro kterou tento jev nastane, musíme nejprve zavést pohybové zkreslení. To je způsobeno příliš dlouhou expozicí pohybujícího se objektu a vzniká i při použití global shutteru. Jako stále přípustné si můžeme zvolit pohybové zkreslení menší než 10%. Doba expozice potom musí být:

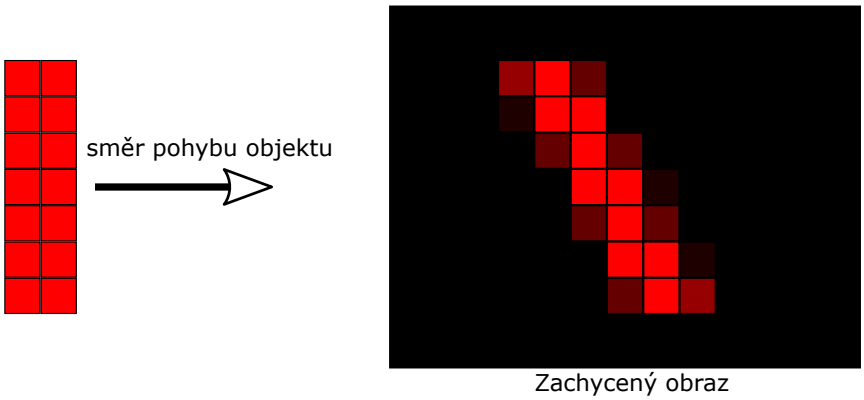
$$T \leq \frac{x}{10v} \tag{2.1}$$

Kde T [μs] je doba expozice celého snímku, x [px] je délka objektu a v [px/μs] je rychlost objektu.

Prostorové zkreslení je potom očekáváno pokud platí:

$$T < y \cdot t_r \tag{2.2}$$

Kde y [px] je výška objektu v pixelech (počet řádků) a t_r [μs] je doba vyčítání jednoho řádku. Lze tedy vidět, že může nastat případ, kdy se při použití rollig shutteru za řádnou cenu nevyhneme zkreslení. Při zvolení delší doby expozice s cílem vyhnutí se prostorovému zkreslení způsobíme pohybové zkreslení a naopak. Sledování rychle se pohybujících objektů může tedy být pro rolling shutter problematické.



Obrázek 2.4. Pohybové zkreslení při snímání rychle se pohybujícího objektu u rollingu shutteru.

Prostorové zkreslení (viz. 2.4) způsobené principem rolling shutteru, může být v některých aplikacích považováno i za výhodu. Při detekci rychlých pohybů by bylo možné nespoléhat se na nezkreslený obraz, ale zaměřit se na hledání fragmentů prostorového zkreslení. Tuto metodu detekce by bylo vhodné použít především pro větší objekty předem známých tvarů, na kterých je prostorové zkreslení nejsnáze pozorovatelné.

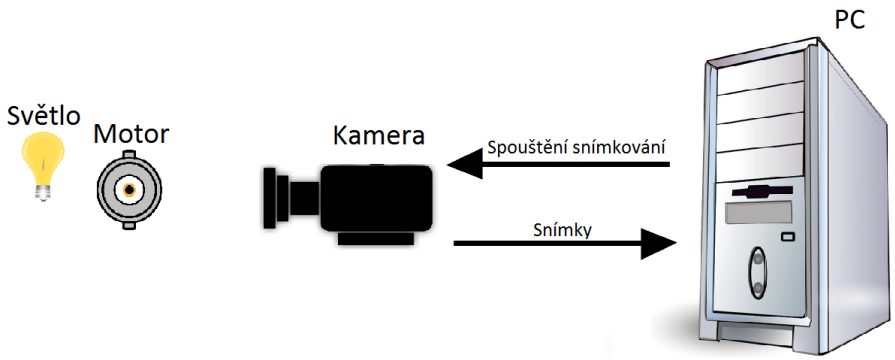
2.3 Návrh měřící sestavy

Po krátké analýze hardwarových možností je důležité si definovat možná řešení celé měřící sestavy. Je téměř jisté, že v jakékoli konfiguraci bude nutné použití přídavného osvětlení. Abychom zamezili pohybovému rozostření, bude se muset doba expozice pohybovat ideálně pod 1 ms. Při takto krátké expozici je velmi obtížné bez přisvětlení nabrat dostatek světla. Přisvětlení nám také zajistí vysoký kontrast snímků velmi vhodný na následující zpracování. Další jisté prvky v měřící soustavě jsou potom kamera, řídící počítač a samozřejmě měřený objekt, jímž bude malý elektromotor. Vibrace budou vyvolávány nevývažkem na hřídeli motoru.

První uvažovaná konfigurace obsahovala navíc ještě objekt upevněný na motor. Mohlo by se jednat například o drát umístěný kolmo k povrchu motoru. Takovýto objekt by na sebe přenášel vibrace motoru a sám kmital. V případě vybrání vhodného objektu, by jeho kmity měly větší amplitudu nežli vibrace motoru. Detekce by tedy byla jednoduší i z důvodu předem známého tvaru objektu. Zároveň bychom ale neměřili vibrace motoru přímo a výpočet změny polohy motoru by se značně komplikoval. Nevýhodou je také nutnost umístění části sestavy na měřený objekt a tím ztracení výhody bezkontaktnosti. Přisvětlení by v této konfiguraci bylo v podobě backlightu.

Další možností je umístění retroreflektoru na motor, který by nahradil zmiňovaný drát z předchozí konfigurace. Backlight by v tomto případě ztratil smysl. Místo toho by bylo použito přímé světlo od kamery směrem na retroreflektor. Zde by docházelo k dorazu světla zpět do kamery. Výsledkem by byly opět velmi kontrastní snímky vhodné pro detekci. Výhody a nevýhody tohoto řešení jsou jinak v podstatě totožné s první možností. Hlavním rozdílem je pouze použití jinak umístěného světla

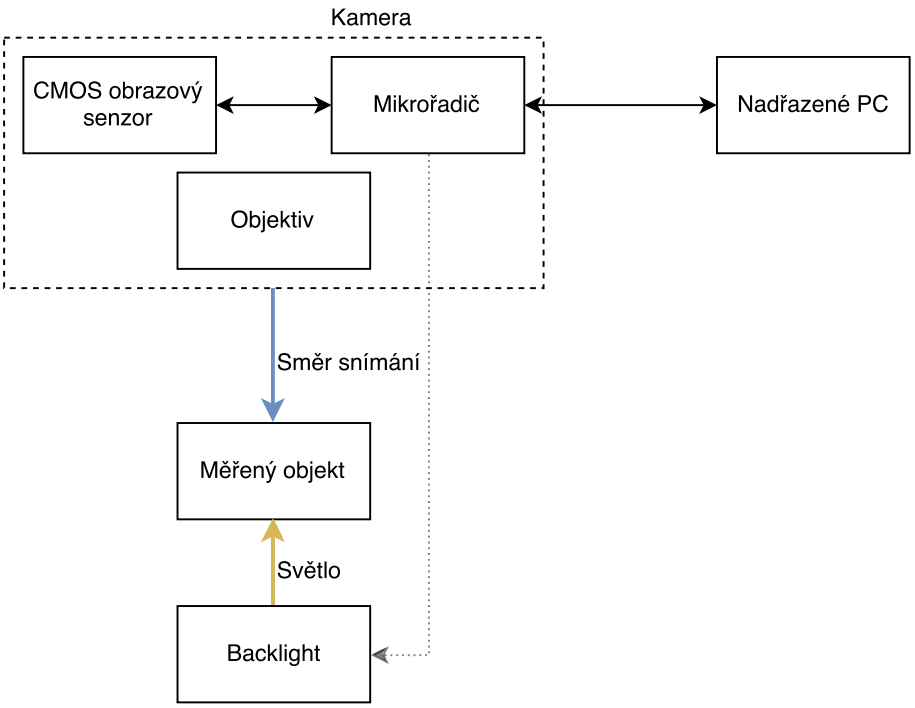
Nakonec byla zvolena nejpřímočařejší možná konfigurace. V ní je snímána přímo hrana objektu. Přisvětlení je v tomto případě nutné použít ve formě backlightu. Hlavní nevýhodou je složitější zpracování nasbíraného obrazu. Toto řešení si ale na rozdíl od ostatních zachovává bezkontaktnost a s tím i větší univerzálnost. Vzhledem k tomu, že je počítáno s použitím ROI, by bylo by možné sledovat sekvenčně více objektů nebo více částí jednoho objektu bez nutnosti pohybu kamery. Ostatní možnosti by v tomto případě byly limitovány nutností umístění dalšího objektu na měřený objekt, což například u rotačních částí strojů není možné. Jednoduché schéma vybrané konfigurace je zobrazeno na následujícím obrázku 2.5.



Obrázek 2.5. Jednoduché schéma měřící sestavy.

Kapitola 3
Realizace měřící sestavy

Tato kapitola se zabývá konkrétním hardwarovým řešením měřící sestavy. Měřící sestavu můžeme rozdělit na čtyři základní části. První částí je nadřazený počítač, ve kterém probíhá veškeré zpracování obrazu. Zároveň také řídí spouštění odběru snímků a komunikaci s kamerou. Druhou částí je kamera, kterou můžeme ještě rozdělit na 2 samostatné části. Kamera je tvořena mikrořadičem a samotným CMOS obrazový senzorem. Mikrořadič řídí CMOS obrazový senzor a komunikuje s nadřazeným počítačem. Další částí je již několikrát zmiňované přisvětlení. Konkrétně je použito v podobě backlightu. V základní verzi není přisvětlení žádným způsobem řízeno. Do budoucna by bylo možné přidat řízení pomocí mikrořadiče v podobě spuštění jen na dobu expozice snímku. Poslední částí je samotný měřený objekt, jehož vibrace mají být detekovány. Jakožto měřený objekt je v práci využit malý elektromotor. Vibrace jsou na něm vyvolávány pomocí nevyváženého zatížení hřídele. Blokové schéma celé sestavy můžeme vidět na obrázku 3.1.



Obrázek 3.1. Blokové schéma měřící sestavy.

3.1 Vývojový kit STM32F429i-Discovery

Základ celého řešení tvoří vývojový kit STM32F429i-Discovery od firmy STMicroelectronics. Tento kit byl pro počáteční vývoj vybrán hned z několika důvodů. Především se již v laboratoři videometrie několikrát osvědčil (včetně několika BP či DP, např. [4]) a existuje tedy propojovací deska mezi kit a použitý CMOS senzor. Další důležité vlastnosti, které nepatří do standartní výbavy vývojových kitů je SDRAM o velikosti 64 Mbit a řadič USB OTG s konektorem USB microAB, fungující bohužel pouze v rychlosti Full-speed (12 Mbit/s). Výhodou je také možnost využití HAL (Hardware abstraction layer) knihoven.

Jako další zajímavé vlastnosti kitu můžeme jmenovat například:

- 32 bitový jednojádrový procesor ARM Cortex-M4 pracující na frekvenci až 180 MHz
- Pouzdro procesoru se 144 piny
- 2 MB Flash paměti
- 256 kB RAM
- 2.4' QVGA (320x240 px) barevný dotykový display
- Podpora DCMI ze strany kontroléru
- Sběrnice I^2C

Samozřejmostí je potom přítomnost věcí jako uživatelské a RESET tlačítko, LED, debugger ST-LINK/V2, DMA, množství čítačů, headery s konektory atd.

Pro shrnutí můžeme ještě jednou vyjmenovat klíčové vlastnosti vybraného kitu pro naši aplikaci:

- 64 Mbit (8 MB) SDRAM
- Full-speed USB
- Podpora DCMI
- Sběrnice I^2C
- DMA (i multibuffer režim)
- Existující podklady v podobě předchozích prací a hotových propojovacích desek
- Možnost využití HAL knihoven

3.2 CMOS senzor Aptina MT9V034

Jako obrazový snímač byl zvolen senzor MT9V034 od firmy Aptina (nyní ON Semiconductor). Jedná se o monochromatický CMOS senzor o rozlišení 752x480 px (WVGA). Maximální frekvence hodinového signálu je 27 MHz a maximální snímkovací frekvence je 60 fps. Senzor používá závěrku typu global shutter. Bylo rozhodnuto použít raději toto přímočařejší řešení místo rolling shutteru 2.2.5. Neméně důležitá je i možnost senzor provozovat v režimu Snapshot. To nám umožní přesně řídit okamžik expozice.

Monochromatická verze senzoru byla zvolena, protože „barevná“ data by nám přinesla pouze složitější práci z daty a žádné výhody. Tento senzor byl už také mnohokrát v laboratoři videometrie používán a existuje tedy PCB modul s tímto senzorem (návrh byl součástí diplomové práce *Synchronizované obrazové snímače* [5]). Výhodná je také existence knihovny pro tento senzor vytvořená v rámci bakalářské práce *Zpracování obrazu mikrořadiči pro mnohokanálové měření polohy* [6], jejíž základ je v práci použit. Další faktorem, který z tohoto senzoru dělá vhodnou volbu je i cena, která se pohybuje přibližně okolo 15 $ (cena za samotný senzor).

Za hlavní důvody výběru tohoto senzoru můžeme tedy považovat:

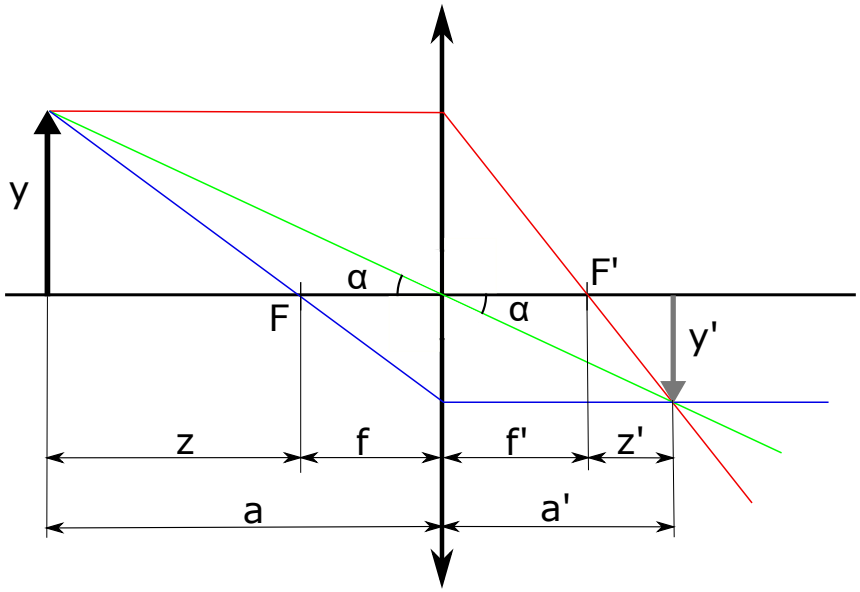
- Global shutter
- Možnost pracovat v Snapshot módu
- Existující PCB modul
- Osvědčené řešení (existující knihovna pro senzor)
- Cena

Mechanické propojení senzoru a kitu potom zajišťuje speciální propojovací deska, která zároveň obsahuje napěťový regulátor (5 V $\rightarrow$ 3 V) pro napájení CMOS senzoru. Tato deska byla vytvořena v rámci diplomové práce *STM32F429 Based CMOS Camera for Teaching Purposes* [7] a je v této práci rovněž používána.

3.3 Objektiv

Nejprve byl použit velmi malý objektiv s ohniskovou vzdáleností $f = 3.6$ mm. U něj se ovšem narazilo hned na několik problémů. Hlavním z nich bylo ostření. S tímto objektivem téměř nebylo možné dostatečně zaostřit na požadovanou hranu. Také samotné ostření je u tohoto objektivu nepraktické vzhledem k nutnosti použití šroubováku. Nedostatek světla je jeden z problémů, se kterým je nutno se vypořádat, a objektiv této velikosti nutně nemůže mít příliš velkou světelnost.

Z těchto důvodů byl vybrán objektiv s výrazně lepšími vlastnostmi. Jedná se objektiv Carl Zeiss Jena DDR Tevidon se světelností f/1.8 a ohniskovou vzdálenost $f = 50$ mm. Disponuje i nastavitelnou clonou, která ale nebude kvůli zachování nejvyšší světelnosti používána. Objektiv je schopný zaostřit nejblíže na objekty vzdálené 500 mm od předmětového ohniska. Je žádoucí tuto hodnotu ještě snížit. Toho je možné dosáhnou pomocí distančního kroužku mezi senzor a objektiv.



Obrázek 3.2. Zobrazení tenkou čočkou.

Závislost mezi ohniskovou vzdálenost f [mm] (uvažujeme-li stejný index lomu v předmětovém i obrazovém prostoru $\Longrightarrow$ předmětová f a obrazová f' ohnisková vzdálenost je stejná), vzdáleností předmětu od předmětového ohniska z [mm] a vzdálenost obrazu od obrazového ohniska z' [mm] můžeme zapsat pomocí Newtonovy zobrazovací rovnice pro tenkou čočku [2]:

$$f^2 = z \cdot z' \tag{3.1}$$

Rovnice (3.1) vychází z obrázku 3.2 a je z ní jasně vidět, že pokud z poroste, tak z' bude klesat. To samé platí samozřejmě i v opačném případě. Naším cílem je nalézt novou minimální hodnotu z, kterou označíme jako z_{new}. Analogicky potom označíme maximální hodnotu z' jako z'_{max}. Výšku distančního kroužku umísťovaného mezi čočku a senzor potom označíme jako l [mm]. Upravenou Newtonovu zobrazovací rovnici potom zapíšeme následovně:

$$f^2 = z_{new} \cdot (z'_{max} + l) \tag{3.2}$$

Protože známe pouze hodnotu f a z_{min}, což je minimální vzdálenost, na kterou je objektiv schopen zaostřit, musíme si ještě hodnotu z'_{max} vyjádřit pomocí známých údajů:

$$z'_{max} = \frac{f^2}{z_{min}} \tag{3.3}$$

Po dosazení a vyjádření z_{new} dostaneme finální vztah pro výpočet nové minimální vzdálenosti ostření objektivu:

$$z_{new} = \frac{f^2 \cdot z_{min}}{f^2 + (l \cdot z_{min})} \tag{3.4}$$

Po dosazení parametrů použitého objektivu $f = 50$ mm, $z_{min} = 500$ mm a výšky distančního kroužku $l = 5$ mm získáme novou minimální vzdálenost předmětu od předmětové ohniska $z_{new} = 250$ mm.

3.4 Kompletní sestava kamery

Sestava tvořící „chytrou“ kameru se tedy skládá z vývojového kitu STM32F429, který řídí CMOS senzor MT9V034 osazený na PCB modulu. Oba prvky jsou potom propojeny speciální deskou umožňující napájení CMOS senzoru buď přímo z kitu nebo z externího zdroje. Tyto jednotlivé prvky by společně měly zajistit všechny požadavky na kameru. Fotodokumentaci jednotlivých prvků sestavy lze nalézt v příloze C.

3.5 Měřený objekt - DC motor

Jakožto měřený objekt byl zvolen malý DC motor. Důvodů pro jeho volbu je hned několik. Velmi snadno na něm vyvoláme vibrace, které budeme následně detekovat. Přidáním nevyvážené zátěže na hřídel začne motor po roztočení vibrovat. Změnou napětí napájecího zdroje potom měníme rychlost otáčení a tím i frekvenci a amplitudu vibrací.

Možnost měnit dva důležité parametry vibrací v podobě amplitudy a frekvence se velmi hodí a dobře prověří schopnosti výsledné měřící sestavy. Vyvolávané vibrace jsou také natolik malé, že dobře ověříme citlivost detektoru. Nespornou výhodou je také velikost elektromotoru, která je pro testování mnohem praktičtější než měření na velkém průmyslovém stroji.

Kapitola 4
Komunikace mezi prvky měřící sestavy

Komunikaci jako celek můžeme rozdělit na dvě hlavní části. Konkrétně potom na komunikaci mezi vývojovým kitem a CMOS senzorem a mezi vývojovým kitem a řídícím počítačem. Tyto dvě části spolu nemají téměř nic společného a jsou řešeny separátně. Komunikace mezi vývojovým kitem a senzorem probíhá čistě na úrovni FW kitu. Druhá část v podobě komunikace vývojového kitu a řídícího počítače je logicky řešena, jak ve FW kitu tak na aplikační úrovni v počítači.

4.1 Komunikace mezi vývojovým kitem a senzorem

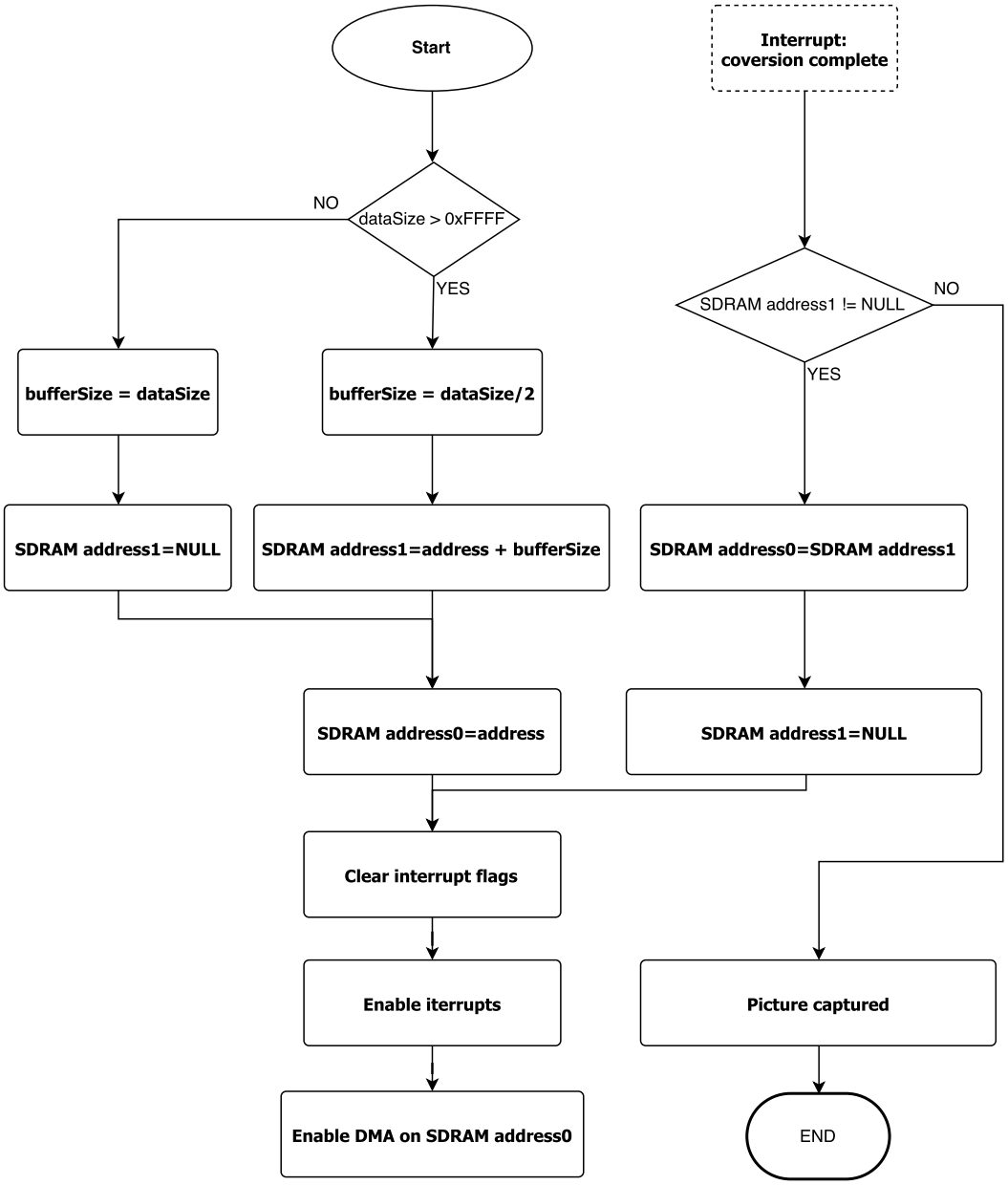
Komunikaci mezi vývojovým kitem a CMOS senzorem můžeme rozdělit do dvou části. První části je přenos obrázku do paměti SDRAM. Tento přenos probíhá pomocí DMA přes paralelní rozhraní DCMI. Použití DMA zaručí velmi rychlý přenos dat přímo do paměti bez zatěžování procesoru. Druhá část komunikace potom probíhá přes sériovou sběrnici I^2C. Ta se zabývá zápisem a čtením konfiguračních registrů senzoru.

4.1.1 Přenos snímků do SDRAM

Jak již bylo zmíněno, přenos snímků do paměti SDRAM probíhá přes rozhraní DCMI pomocí DMA. DCMI je speciální paralelní rozhraní firmy STmicroelectronics určené pro práci s digitálními obrazovými senzory. Jeho využití k propojení CMOS senzoru a vývojového kitu tedy byla jasná volba. Stejně jako senzor lze DCMI provozovat v režimu Snapshot. V tomto módu je načtení snímku řízeno pomocí Capture bitu v konfiguračním registru periferie. Po jeho nastavení čeká DCMI na první FrameValid signál od senzoru a následně načte snímek. Capture bit je po načtení snímku automaticky vynulován a další načtení snímku do paměti se provede znovu až po uživatelem nastavení Capture bitu.

Samotný přenos snímků do paměti SDRAM vývojového kitu zajišťuje DMA. Podpora DMA ze strany DCMI a tedy možnost jeho využití přináší hned několik výhod. Jedna z nich plyne z funkčnosti DMA. Po nastavení periferie běží přenos dat do paměti bez asistence procesoru, který může ve stejný okamžik vykonávat jiné akce. Výhodná je také rychlost DMA. Ta se může pohybovat až kolem hodnot 54 MB/s. Přenos snímku při využití ROI o velikosti 180x60 px bude trvat přibližně 200 μs, což je rychlost, které bychom jiným způsobem těžko dosáhli.

Při přenosu snímku plné velikosti 752x480 px ovšem nastává problém. Dle dokumentace [8] lze při použití jednoho bufferu přenést 65 535 datových jednotek. Vzhledem k tomu, že DMA podporuje pouze 32-bitový přístup, můžeme přes jeden buffer přenést 262 140 bytů. Snímek plné velikost má ovšem bytů 360 960. Je tedy nutné provést přenos snímku nadvakrát. To vyžaduje dvě nastavení DMA, kdy druhé následuje po dokončení prvního přenosu. DMA periferie umožňuje uložit do konfiguračních registrů dvě paměťové lokace, na které je možné data přenášet. Snímek je před zahájením přenosu rozpůlen a do konfiguračních registrů DMA jsou nahrány odpovídající paměťové lokace. Nejprve je zahájen přenos snímku do lokace 1. Po jeho dokončení je lokace 1 nahrazena lokací 2 a je spuštěn přenos druhé části obrazu. Celý proces přenosu snímku do SDRAM je zachycen na vývojovém diagramu 4.1.



Obrázek 4.1. Vývojový diagram konfigurace přenosu snímku do SDRAM pomocí DMA.

DMA periferie umožňuje i střídavé zapisování na nakonfigurované lokace v tzv. multi buffer módu. Vzhledem k proměnné velikosti přenášených dat, by ale bylo nutné přepínat režim DMA mezi single a multi buffer módem. Za předpokladu, že většinou budou přenášena data menší než 262 140 bytů, se nevyplatí řešit problematiku přepínání módů a věcí s tím souvisejících. Reálné zrychlení by pro naší aplikaci bylo nulové, protože doba mezi snímky je výrazně delší než doba přenosu snímku do paměti. Využití multi buffer módu leží především v opětovném střídavém zápisu do nastavených lokací se zpracováním dat během přenosu do druhé lokace. V naší aplikaci je nutné data pouze přenést a rychlost single buffer režimu je více než dostačující.

4.1.2 Čtení a zápis do konfiguračních registrů senzoru

Přes rozhraní DCMI je možné vyčítat snímky, ale neumožňuje přístup ke konfiguračním registrům senzoru. Přístup k nim je řešen pomocí sběrnice I^2C. Ta nám umožňuje zápis i čtení konfiguračních registrů. Díky tomu jsme především schopni nastavovat senzor do požadované konfigurace. Po přijmutí požadavku ze strany řídícího počítače, je příkaz zpracován a následně provedena požadovaná akce přes I^2C sběrnici.

I^2C je velmi hojně využívaná sériová sběrnice. Funguje na principu Master-Slave. Master generuje hodinový signál a iniciuje komunikace. Slave hodinový signál přijímá a po adresaci reaguje. Na sběrnici se může vyskytovat v jednom okamžiku několik Masterů a jednotlivé zařízení mohou své módy měnit. V našem případě je Master řídící mikroprocesor a Slave CMOS senzor.

4.2 Komunikace mezi vývojovým kitem a řídícím PC

Komunikace mezi vývojovým kitem a řídícím počítačem probíhá přes sběrnici USB. Na úrovni FW ve vývojovém kitu je využito USB CDC knihovny. USB CDC je standardizovaná USB třída umožňující simulování různých komunikačních rozhraní na úrovni FW. Nejběžněji se využívá simulace COM-portu (Virtual COM-port) a Ethernetu (Ethernet-over-USB). V našem případě bylo využito simulace COM-portu kvůli jednoduchosti práce s tímto portem. Jelikož je USB CDC standardizováno, měla by Plug-and-Play funkcionalita fungovat na všech běžných operačních systémech (Windows, Linux, Mac OS). Po připojení k PC se zařízení připojí jakožto Sériové zařízení USB a můžeme k němu přistupovat přes COM-port. Oproti fyzickému COM-portu má však jistá omezení. Samozřejmě nelze využívat Request-to-Send hardwarové řízení toku. Bohužel se mi nepodařilo potvrdit ani funkčnost XON/XOFF softwarového řízení toku.

4.2.1 Řízení toku dat mezi vývojovým kitem a řídícím PC

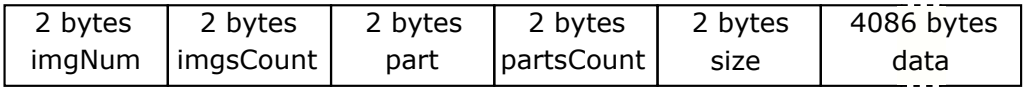
Bez jakéhokoli řízení toku docházelo při odesílání většího množství dat náhodně ke ztrátě části dat. Jediná velká data posílána z vývojového kitu do PC jsou snímky. Pokud došlo ke ztrátě dat při odesílání snímku bylo nutné opakovat přenos celého snímku, protože nebyla k dispozici informace o tom, která data se ztratila. To mohlo vyústit až k velmi znatelnému zvýšení doby poslání potřebných dat.

Z analýzy problému vyplynulo, že problém s největší pravděpodobností není na straně vývojového kitu. Vše ukazovalo na to, že data se skutečně odešlou. Problém tedy musel být na straně řídícího PC. Vše ukazovalo na to, že je tento problém způsoben již na úrovni implementace ovladačů Virtual COM-portu, protože stejný problém nastával jak v komunikaci s aplikací psanou v jazyku C# tak v Matlabu. Na druhou stranu docházelo ke ztrátě dat i při použití dvou různých ovladačů. Výchozí volbou ovladače v systému Windows je generický ovladač dodávaný přímo Microsoftem. Pokud je zařízení ručně instalováno, lze jej spustit i přes ovladač Virtual COM-portu od STMicroelectronics. Jelikož byly výsledky stejné s oběma ovladači, rozhodl jsem se místo hlubší analýzy problém řešit.

Jedním možným řešením bylo pokusit se zpomalit odesílání dat. Uvažovaným problémem bylo příliš rychlé odesílání dat, kdy se data nestíhala v nějaký okamžik vyčítat z interního bufferu Virtual COM-portu. To mohlo potom vyústit k přetečení tohoto bufferu. Zpomalením odesílání dat by potom aplikace na straně řídícího počítače získala více času na vyčítaní dat a zamezilo by se přetékání bufferu. Po zpomalení odesílání dat, pomocí přidání prodlevy mezi odesíláním jednotlivých bloků dat, skutečně došlo ke zlepšení. Již při zpomalení na přibližně 700 kB/s docházelo ke ztrátě dat podstatně méně. Teprve

při zpomalení na zhruba 500 kB/s se však dalo považovat komunikaci za dostatečně spolehlivou. Ukázalo se ale, že zpomalení odesílání dat je velmi neflexibilní řešení. Rychlost přijímání dat totiž závisí i na výkonu řídícího PC a dostupném procesorovém času pro aplikaci přijímající data. Při změně zatížení PC nebo puštění na jiném PC potom znovu docházelo ke ztrátě dat. Bylo by možné odesílání ještě zpomalit, ale stále by to nebylo jisté řešení a rychlost by se už zásadně snížila.

Rozhodl jsem se tedy obrázky odesílat po paketech s pevně danou délkou. Řídící počítač si nejprve musí vyžádat odeslání snímku a následně i jednotlivých paketů. FW odešle další paket až po zpracování požadavku o jeho odeslání. Řídící počítač si také může vyžádat přeposlání posledního paketu v případě, že by nastal problém. Takto bylo vytvořeno řízení toku jednotlivých paketů na aplikační úrovni. Ukázalo se, že toto řešení je již spolehlivé a zamezuje ztrátě dat. Při běžném provozu není ani třeba přeposílání jednotlivých paketů, ale pro případ jiného problému byla tato funkcionalita zachována. Není také nutné jiným způsobem omezovat rychlost odesílání dat jako v předchozím případě.



Obrázek 4.2. Nákres paketu pro odesílání snímků.

Offset	Počet bytů	Datový formát	Popis
0	2	uint16	Číslo snímku v paměti
2	2	uint16	Celkový počet snímků v paměti
4	2	uint16	Číslo paketu v rámci snímku
6	2	uint16	Počet paketů pro snímek
8	2	uint16	Počet bytů obsahující užitečná data
10	4086	byte[]	Data

Tabulka 4.1. Popis částí paketu pro odesílání snímků.

4.2.2 Rychlost přenosu dat mezi vývojovým kitem a řídícím PC

Pro přenos dat mezi vývojovým kitem a řídícím PC je použito full-speed USB (USB 1.1). Teoretická maximální rychlost přenosu dat je tedy 12 Mbit/s, neboli 1.5 MB/s. Efektivní reálná rychlost však dosahuje maximálně 1 MB/s. To je způsobeno především režií USB přenosu jako takového. Při využití Virtual COM-portu se efektivní rychlost ve většině případů ještě o něco málo sníží. Pokud by se tedy podařilo dostat rychlost odesílání dat z vývojového kitu do PC na hodnoty okolo 900 kB/s, mohli bychom to považovat za úspěch.

Rychlost odesílání/přijímání dat má smysl měřit pouze ve směru vývojový kit řídící počítač, protože jedině zde se přenáší větší množství dat ve formě nasnímaných obrázků. Před zavedením paketů se efektivní rychlost odesílání pohybovala přibližně okolo 500 kB/s (měřena doba odeslání celé paměti i s nutností přeposílání dat). Po implementaci odesílání po paketech a řízení toku na aplikační úrovni se rychlost odesílání znatelně navýšila až k hodnotám okolo 800 kB/s.

Kvůli způsobu implementace se ale rychlost výrazně mění s velikostí odesílaných snímků. Protože jsou snímky odesílány postupně a ne jako jeden blok dat, je se zvyšujícím se počtem snímků odesíláno větší množství neužitečných dat. Pokud už zbývá odeslat ze snímku menší množství dat než je velikost paketu, je i tak odeslán paket plné velikosti s daty,

která jsou následně zahozeny. Takovýto způsob implementace sráží rychlost přenosu, ale na druhou stranu je velmi přehledný pro napsání aplikace komunikující s FW. Při odesílání snímků plné velikost (752x480 px) se efektivní rychlost pohybuje kolem maximální hodnoty 800 kB/s. Pokud ale snížíme velikost snímků například na hodnotu 160x80 px efektivní rychlost se sníží na 500 kB/s. Jestliže vezmeme v potaz velikost paketu a velikost obrázku zjistíme, že téměř 1/4 odesílaných dat jsou neužitečná data. Po započtení zpomalení v aplikaci způsobené nutností zpracovaní většího množství individuálních obrázků se dostaneme přibližně na onen rozdíl v rychlosti 300 kB/s. Nejhorší možný případ by bylo odesílání snímků o velikosti málo větší než velikost paketu. Potom by byla odesílána téměř 1/2 neužitečných dat. Rychlost by se v tomto případě pravděpodobně dostala pod 400 kB/s.

Jak je vidět použité řešení určitě není ve všech směrech ideální. Jelikož se ale jedná především o testovací aplikaci, je výhodné prozatím ponechat jednoduché a jasné řešení. Optimalizaci lze provést po zjištění ideální konfigurace celé sestavy. Nejjednodušší by bylo optimalizovat velikost paketu vzhledem k velikosti odesílaných obrázků. Ve FW se v zásadě jedná o přepsaní jedné definované hodnoty. Druhou a pravděpodobně ještě efektivnější možností by bylo místo posílání jednotlivých snímku (při odesílání celé sekvence) odeslat celou paměť SDRAM jako jeden blok dat. Toto řešení by ale značně změnilo způsob zpracování dat na straně PC aplikace a vyžadovalo větší zásahy do kódu.

4.2.3 Dostupné příkazy pro kameru

Komunikace směrem od řídícího počítače k vývojového kitu se skládá především z příkazů k začátku snímkování, nastavování senzoru a požadavků o odeslání snímků. Druhým směrem (tedy od vývojového kitu k řídícímu počítači) jsou to především nasnímané obrázky a pomocné potvrzovací příkazy o dokončení snímání.

V následujícím výpisu jsou zaznamenány a popsány všechny dostupné příkazy pro kameru. Všechny příkazy jsou tvořeny skupinou ASCII znaků ukončenou znakem '\n'.

1. **Příkazy pro snímání**

   - `cs\n`
     Zahájí sejmutí jednoho snímku dle aktuálního nastavení senzoru a jeho následné uložení do paměti SDRAM. Snímek vytvořený po tomto příkazu je vždy uložen na začátku paměti. Získává tedy v paměti číslo 0. Po dokončení přenosu snímku do SDRAM FW odešle jeden potvrzovací byte o hodnotě `uint8 confirm = 1`.
   - `ccXX\n`
     Zahájí snímání sekvence snímků na stroboskopické frekvenci odpovídající parametru `XX`. Parametr `XX` je frekvence v Hz. Povolený rozsah zadávané frekvence je v intervalu $\langle 1, 99 \rangle$ Hz. Výsledkem je maximální množství snímků, které je možné uložit do SDRAM, nasnímaných na vypočtené stroboskopické frekvenci s hustotou 20 snímků na periodu zadané frekvence. Snímky jsou do paměti ukládány postupně v pořadí v jakém byly pořízeny. Po dokončení přenosu posledního snímku do SDRAM FW odešle jeden potvrzovací byte o hodnotě `uint8 confirm = 1`.

2. **Příkazy pro odesílání snímků**

   - `dsXXXX\n`
     Připraví FW na odesílání snímku s číslem v paměti odpovídajícím parametru `XXXX`. Parametr `XXXX` je číslo snímku v paměti. Rozsah tohoto parametru se mění podle velikosti snímků odpovídající momentálnímu nastavení senzoru. Není tedy možné odesílat snímky jiné velikosti než je momentální nastavení senzoru. Adresa do SDRAM od které se začnou požadovaná data odesílat je vypočtena

jako: `addressToSend = SDRAMBankAdd + XXXX · imageSize`, kde `SDRAMBankAdd` je adresa začátku SDRAM, `XXXX` je číslo obrázku v paměti (převedeno do číselného datového typu) a `imageSize` je velikost obrázku v bytech. Pokud je parametr `XXXX` zadán mimo momentální rozsah, FW zůstane ve výchozím stavu a vůbec se nepřepne do stavu odesílání. Další příkazy pro odesílání tedy nebudou pertinentní. Je důležité zmínit, že tento příkaz nezahájí samotné odesílání. Pouze připraví FW na odesílání požadovaného snímku. Použití příkazů pro odesílání snímků ze strany řídícího PC je zobrazeno na obrázku 4.3.

- `dn\n`
  Pokud je FW ve stavu odesílání snímku nastaveném příkazem `dsXXXX\n` zahájí odesílání dalšího paketu vybraného snímku. Po odeslání posledního paketu je FW přepnut do výchozího stavu.
- `dr\n`
  Pokud je FW ve stavu odesílání snímku nastaveném z příkazu `dsXXXX\n` zahájí opakované odeslání předchozího odeslaného paketu.

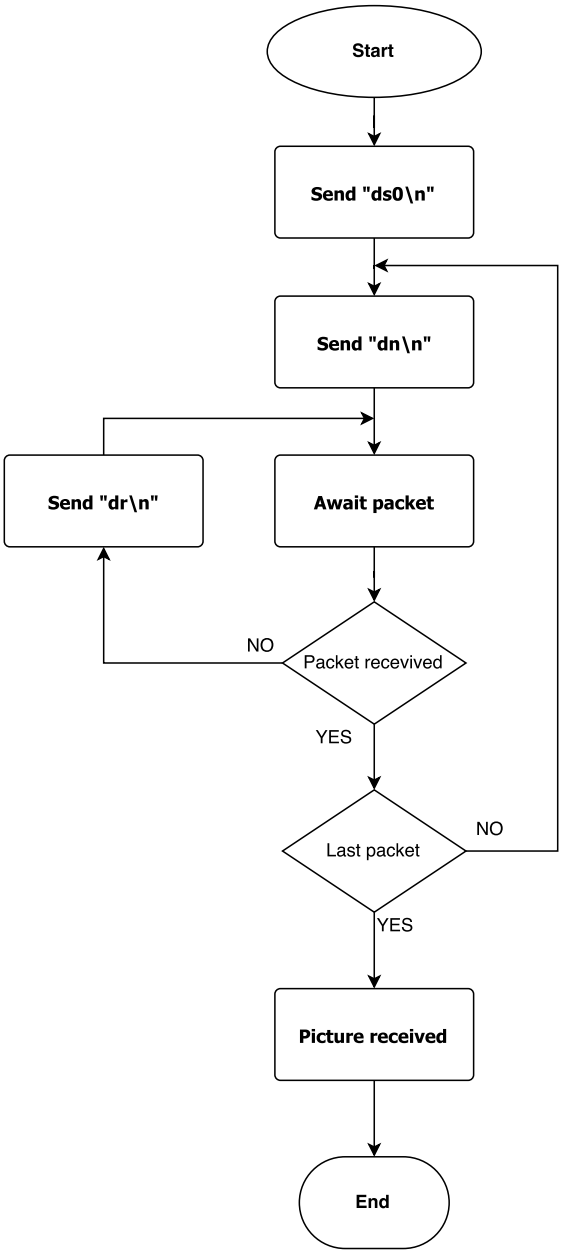
3. **Příkazy pro nastavování senzoru**

- `sagX\n`
  Příkaz pro nastavení AGC (Automatic Gain Control). Parametr `X` může být buď '1' pro zapnuto nebo '0' pro vypnuto. Pokud je AGC zapnut, hodnota analogového zesílení senzoru je nastavována automaticky a hodnota nastavená uživatelem tedy není pertinentní.
- `saeX\n`
  Příkaz pro nastavení AEC (Automatic Exposure Control). Parametr `X` může být buď '1' pro zapnuto nebo '0' pro vypnuto. Pokud je AEC zapnut, doba expozice je nastavována automaticky a hodnota nastavená uživatelem tedy není pertinentní.
- `seXXXX\n`
  Příkaz pro nastavení doby expozice. Parametr `XXXX` je požadovaná doba expozice v μs. Povolené jsou hodnoty v intervalu $\langle 100, 9999\rangle$ μs. Tuto hodnotu ale není možné nahrát přímo do senzoru. Je nutné ji přepočítat na dobu expozice v počtu řádků. Po zpětném přepočtu se kvůli zaokrouhlování na celá čísla může nastavená hodnota lišit od požadované maximálně o 30 μs. Takto nastavená hodnota doby expozice je pertinentní pouze v případě, že AEC je vypnut.
  Vzhledem k tomu, že v této aplikaci je požadované především, aby se doba expozice pro sekvenci snímku neměnila, je přesnost pomocí tohoto příkazu dostačující. Je ovšem vhodné zmínit možnost dosáhnutí větší přesnosti nastavování doby expozice pomocí pomocného konfiguračního registru senzoru. S jeho použitím je možné dosáhnou přesnosti až okolo 50 ns.
- `sgXX\n`
  Příkaz pro nastavení velikosti analogového zesílení. Parametr `XX` je požadovaná hodnota analogového zesílení. Povolené jsou hodnoty v intervalu $\langle 16, 64\rangle$, 16 odpovídá minimálnímu zesílení 1x a 64 maximálnímu zesílení 4x. Takto nastavená hodnota je pertinentní pouze v případě. že AGC je vypnut.
- `sbXX\n`
  Příkaz pro nastavení jasu snímku. Parametr `XX` je požadovaná jasnost. Povolené jsou hodnoty v intervalu $\langle 1, 64\rangle$, 1 odpovídá minimálnímu jasu a 64 maximálnímu jasu. Takto nastavená hodnota je pertinentní pouze v případě. že AEC i AGC je zapnuto.

- `ss\n`
  Příkaz vyvolá nahrání parametrů o velikosti snímku uložených v konfigurační struktuře do konfiguračních registrů senzoru. Pro úspěšné nahrání musí být splněna následující podmínka: `width + startX < 753 && height + startY < 481`.
- `swXXX\n`
  Příkaz pro nastavení šířky snímku do konfigurační struktury. Parametr `XXX` je požadovaná šířka snímku v pixelech. Povolené jsou hodnoty v intervalu $\langle 1, 752 \rangle$. Příkaz nenastavuje změnu do konfiguračního registru senzoru! Samotné nahrání je následně nutné potvrdit příkazem `ss\n`.
- `shXXX\n`
  Příkaz pro nastavení výšky snímku do konfigurační struktury. Parametr `XXX` je požadovaná výška snímku v pixelech. Povolené jsou hodnoty v intervalu $\langle 1, 480 \rangle$. Příkaz nenastavuje změnu do konfiguračního registru senzoru! Samotné nahrání je následně nutné potvrdit příkazem `ss\n`.
- `sxXXX\n`
  Příkaz pro nastavení x souřadnice začátku snímku do konfigurační struktury. Parametr `XXX` je požadovaná x souřadnice začátku snímku. Povolené jsou hodnoty v intervalu $\langle 0, 750 \rangle$. Příkaz nenastavuje změnu do konfiguračního registru senzoru! Samotné nahrání je následně nutné potvrdit příkazem `ss\n`.
- `syXXX\n`
  Příkaz pro nastavení y souřadnice začátku snímku do konfigurační struktury. Parametr `XXX` je požadovaná y souřadnice začátku snímku. Povolené jsou hodnoty v intervalu $\langle 0, 478 \rangle$. Příkaz nenastavuje změnu do konfiguračního registru senzoru! Samotné nahrání je následně nutné potvrdit příkazem `ss\n`.

4. **Příkaz pro odeslání obsahu konfiguračních registrů**

- `r\n`
  Příkaz vyvolá načtení konfiguračních registrů přes I^2C sběrnici a následně jejich hodnoty odešle jako bytové pole.



Obrázek 4.3. Vývojový diagram přijímání snímku ze strany řídícího PC.

Kapitola 5
Snímání obrazu

Způsob odběru jednotlivých snímků je klíčová část této práce. Nejprve budou stanoveny cíle a teoretické předpoklady řešení. To zahrnuje především hlubší rozbor stroboskopického snímkování a řešení velikosti snímků. Následně bude probráno i konkrétní řešení implementace odběru snímků ve FW vývojového kitu.

5.1 Vstupní požadavky

Jak již bylo zmíněno v úvodu, práce se zabývá detekcí vibrací. Frekvence hledaných vibrací se bude pohybovat přibližně mezi 5-50 Hz. Použitý senzor ovšem zvládá snímkovat pouze v řádu desítek fps. Je proto nutné pro bezpečnou detekci použít stroboskopického snímkování, jemuž bude věnována zvláštní podsekce.

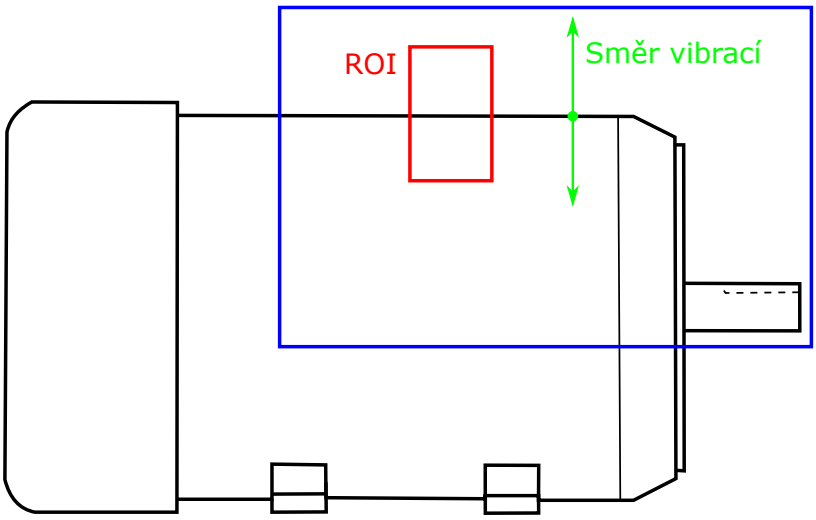
Z rychlosti snímaného pohybu nám přímo vyplývá další požadavek na snímání. Aby obraz nebyl rozmazaný, musíme dodržet určitou hranici doby expozice. V našem případě by neměla překročit hodnotu 1 ms. Při takto malé době expozice je nutné mít dostatek světla. Je tedy nutné přidat přídavné osvětlení za snímaný objekt (ideově viz. obrázek 2.5), tak aby světlo dopadalo do objektivu. Zároveň si tímto způsobem zvýrazníme sledovanou hranu.

Počet snímků, který můžeme udělat při jednom odběru je omezen několika faktory. Především je to velikost SDRAM použitého kitu, která je 8 MB. Odesílání snímků během odběru by nám téměř nepomohlo, protože kit podporuje pouze Full-speed USB. Při jednom odběru si tedy musíme vystačit pouze s velikostí SDRAM. Při snímkování v plném rozlišení senzoru (752x480 px) je velikost jednoto snímku 360960 bytů. Z velikosti SDRAM a velikosti jednoho snímku snadno dopočítáme, že bychom při plném rozlišení mohli udělat sekvenci pouze 23 snímků. To je pro detekci vibrací zoufale málo a je tedy nutné nějakým způsobem zvětšit počet snímků v sekvenci.

Řešením tohoto problému je ROI (Range Of Interest). Díky ROI snížíme velikost snímaného obrazu. Použitý senzor tuto možnost přímo podporuje a do SDRAM je pomocí DMA skutečně přenášena pouze požadovaná část obrazu. V tabulce 5.1 můžeme vidět, že při využití ROI množství snímků pro jednu sekvenci rapidně stoupá. Způsob výběru ROI je zobrazen na obrázku 5.1. Výběr ROI je takový, aby ve výřezu byla zachycena část hrany s dostatečně velkou oblastí kolmo na obě strany. Vzhledem k hledání pohybu pouze v horizontálním nebo vertikálním směru stačí, aby byl výřez kolmo ke směru vibrací široký pouze v řádech několika desítek px.

Rozlišení [px]	Počet snímků
752x480	23
640x480	27
320x240	109
200x100	419
180x60	776

Tabulka 5.1. Porovnání počtu snímků pro naplnění SDRAM při různém rozlišení.



Obrázek 5.1. Princip výběru ROI.

5.2 Stroboskopické snímkování

Stroboskopické snímkování vychází z fenoménu zvaného stroboskopický jev. Jinak se také stroboskopické snímkování nazývá jako snímkování v ekvivalentním čase. Tento název velmi dobře vystihuje princip stroboskopického snímkování. Při snímkování periodického průběhu ve vhodné okamžiky získáme vzorky ekvivalentní vzorkům získaných na mnohem vyšší vzorkovací frekvenci. Díky tomu můžeme úspěšně rekonstruovat signál vzorkovaný na frekvenci odporující vzorkovacímu teorému. Jediným rozdílem je potom časová osa. Tohoto principu je využito například v některých módech vzorkování osciloskopů.

5.2.1 Stroboskopický jev

Jak již bylo několikrát zmíněno, je v tomto konkrétním případě nutné použít stroboskopického snímkování. Tato metoda je založena na takzvaném stroboskopickém jevu. Stroboskopický jev je vlastně speciální případ aliasingu, jehož vznik hrozí při nedodržení vzorkovacího teorému. Ten můžeme pomocí vztahu vyjádřit následovně:

$$f_v > 2f_{max} \tag{5.1}$$

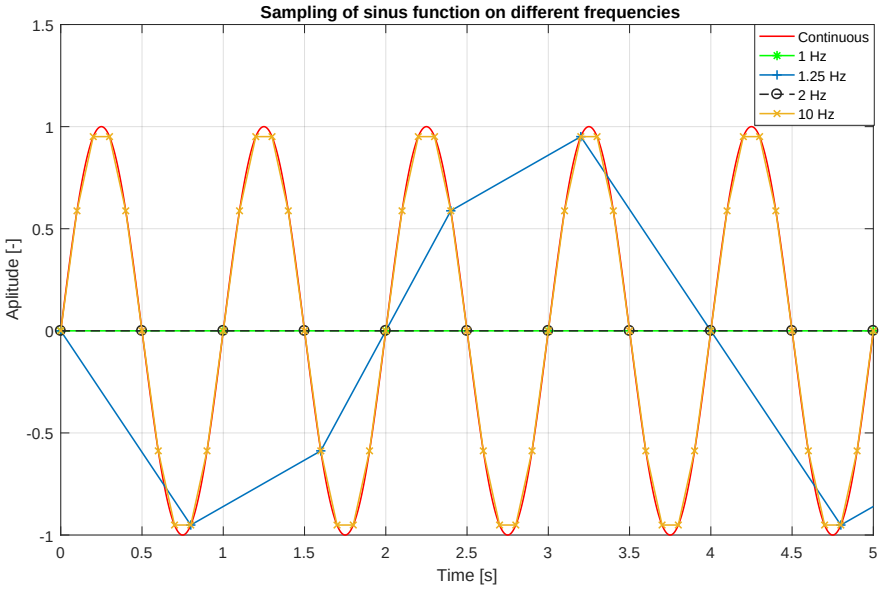
Kde f_v [Hz] je vzorkovací frekvence a f_{max} [Hz] je frekvence nejvyšší harmonické složky v signálu. Slovně tedy můžeme vzorkovací teorém přepsat jako:

„Frekvence vzorkování musí být alespoň dvakrát vyšší než-li frekvence nejvyšší harmonické složky vzorkovaného signálu.“

Podmínkou pro vznik stroboskopického jevu je potom, že frekvence vzorkovaného signálu je celočíselným násobkem vzorkovací frekvence, neboli:

$$n \cdot f_v = f_s \; ; \quad n \in \mathbb{N} \tag{5.2}$$

Kde f_v [Hz] je vzorkovací frekvence a f_s [Hz] je frekvence vzorkovaného signálu a $\mathbb{N}$ je množina přirozených čísel. Ukázku aliasingu a stroboskopického jevu je zachycena na obrázku 5.2.



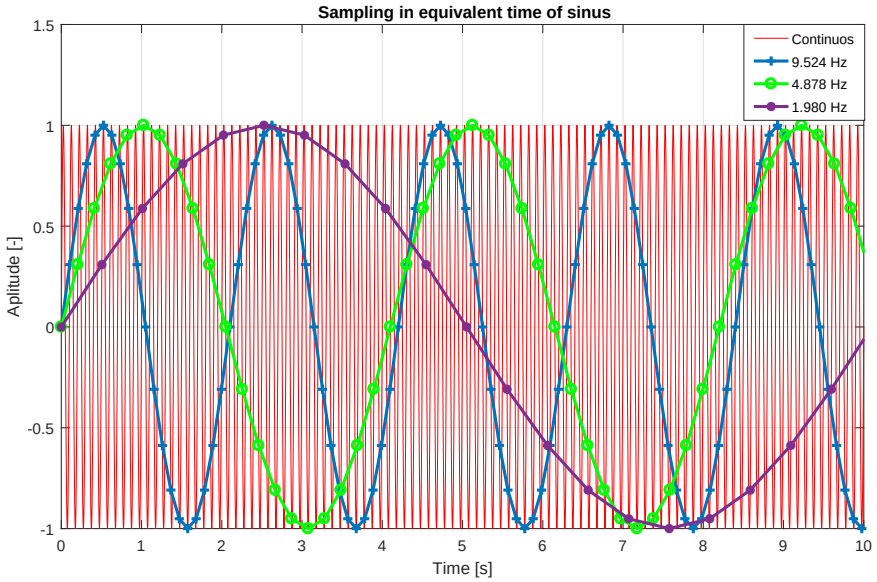
Obrázek 5.2. Ukázka aliasingu na vzorkování sinu o frekvenci 1 Hz.

5.2.2 Možná použití stroboskopického jevu

Nabízejí se hned dvě různá využití stroboskopického jevu. První možností je provedení odběrů sekvencí snímků na několika různých frekvencích. Při postupném snižování snímkovací frekvence dospějeme do bodu, kdy bude platit podmínka (5.2). S dostatečně dlouhou sekvencí snímků zajištěnou díky ROI jsme potom s jistotou schopni říci, že právě na této frekvenci objekt kmitá. Objekt se při takové frekvenci bude jevit jako statický, zatímco na frekvencích nižších i vyšších se bude hýbat. Protože nás ale spíše než frekvence vibrací zajímá jejich amplituda, byla zvolena druhá možnost, a to použití snímkování v ekvivalentním čase (stroboskopického snímkování).

Při stroboskopickém snímkování se nesnažíme o zachycení statického obrazu. Místo zachycení vzorku vždy ve stejný okamžik periody (čas mezi snímky je přesně roven periodě pohybu) se v periodě vždy o malý časový okamžik posuneme. Toho je dosaženo čekáním před dalším odběrem dobu málo vyšší než je perioda pohybu. Hodnotou přičtenou k periodě jsme potom schopni i přesně řídit kolik snímků za periodu bude pořízeno. Výsledný zachycené vibrace jsou ideálně až na měřítko časové základny totožné s hledanými vibracemi. Podmínkou je především přesnost řízení odběru snímků a periodicita pohybu. Tímto způsobem jsme schopni simulovat snímkovací frekvenci až několikanásobně vyšší, než nám umožňuje senzor samotný. Podobného principu je například využito v některých módech vzorkování osciloskopů. Ukázka vzorkování v ekvivalentním čase je zobrazena na obrázku 5.3.

Výhody ovšem nejsou to jediné, co stroboskopické snímkování přináší. Jeho použitím vznikají i jistá omezení. Před začátkem snímkování musíme znát frekvenci vibrací (periodu hledaného pohybu), na které chceme stroboskopicky snímkovat, abychom získali správný periodický průběh pohybu. Jak již také bylo zmíněno, stroboskopické snímkování vyžaduje velmi přesné doby odběru jednotlivých snímků pro dodržení podmínek funkčnosti.



Obrázek 5.3. Ukázka stroboskopického vzorkování na sinu o frekvenci 10 Hz.

5.2.3 Výpočet stroboskopické frekvence

Výpočet stroboskopické frekvence i se zahrnutím požadovaného ekvivalentního počtu vzorků je v zásadě velmi jednoduchý. Nejnázornější je přepočet přes periodu T [s]. Nejprve je vypočítána perioda odpovídající požadované frekvenci snímkování f [Hz]:

$$T = \frac{1}{f} \tag{5.3}$$

Kde T [s] je perioda snímkování a f [Hz] je frekvence snímkování. Následně už můžeme vypočítat periodu ekvivalentního vzorkování:

$$T_{ekv} = n \cdot T + \frac{T}{fpp} = T \cdot \frac{n \cdot fpp + 1}{fpp} \; ; \quad n \in \mathbb{N} \tag{5.4}$$

Kde T_{ekv} [s] je perioda stroboskopického vzorkování a fpp [-] je požadovaný ekvivalentní počet snímků na periodu (frames per period). Jelikož je reálně více důležitá právě perioda T_{ekv} určující dobu mezi snímky, není nutné přepočítávat tuto hodnotu zpět na ekvivalentní frekvenci f_{ekv} [Hz]. Pokud bychom chtěli můžeme vztah mezi fpp, požadovanou frekvencí a ekvivalentní frekvencí vyjádřit následovně:

$$f_{ekv} = f \cdot \frac{fpp}{n \cdot fpp + 1} \; ; \quad n \in \mathbb{N} \tag{5.5}$$

5.3 Implementace snímkování ve vývojovém kitu

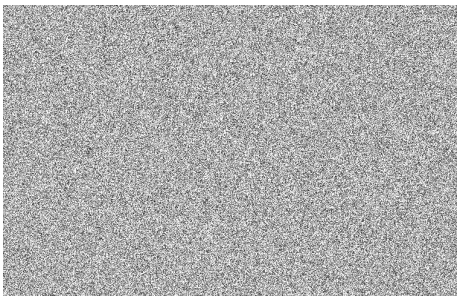
CMOS senzor i DCMI periferie jsou provozovány v režimu snapshot, který umožňuje uživatelem řízené snímkování. K řízení celého procesu si vystačíme pouze se dvěma proměnnými:

- Exposure signál CMOS senzoru
- Capture bit v konfiguračním senzoru DCMI periferie

Začátek expozice je spouštěn exposure signálem senzoru. DCMI periferie je potom řízena pomocí capture bitu v konfiguračním registru. Po jeho nastavení do jedničky se následně čeká na první frame valid, jehož přijetí započne přenos snímku od SDRAM. Frame valid signál indikuje konec expozice a možnost začít vyčítat snímek. Capture bit je automaticky vynulován po dokončení přenosu.

5.3.1 Šum v Snapshot módu senzoru

Při použití Snapshot módu CMOS senzoru se standartě jeho exposure signál po začátku expozice přepíná zpět do nuly, aby se zamezilo další expozici bez zásahu uživatele. Při použití této logiky jsem ovšem narazil na problém. Pokud zůstane exposure signál v nule (vypnutý) delší dobu (chápejme např. déle než 1 s), začne se na senzoru hromadit šum. To i v případě, že má objektiv plně zavřenou mechanickou clonu a navíc je na objektivu krytka. Šum tedy zjevně není světelného rázu a ani se tak na následujícím pořízeném snímku 5.4 nejeví. Na vině je kumulace náboje na senzoru CMOS. Pokud je exposure signál v nule, nedochází totiž vůbec k vyčítání dat ze senzoru. Náboje vznikající na senzoru se tedy mohou kumulovat a projeví se teprve při vyčítání dalšího snímku.

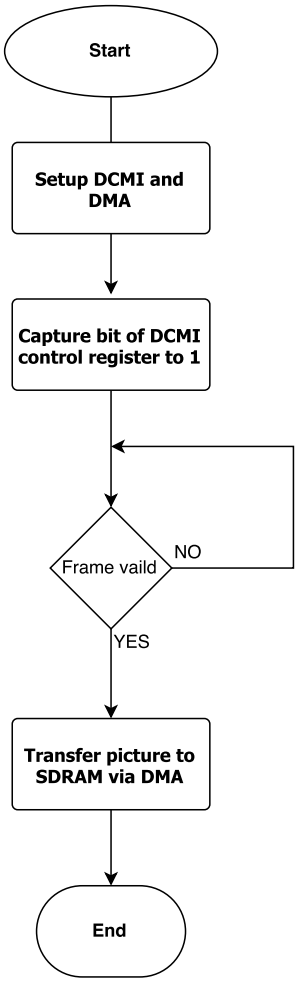


Obrázek 5.4. Ukázka šumu způsobeného kumulací náboje v senzoru CMOS.

Problém byl vyřešen jednoduše tak, že za výchozí stav exposure signálu je považován stav v jedné (zapnuto). V tomto případě senzor pracuje jako ve standardním kontinuálním módu. Snímky jsou tedy pořizovány maximální rychlostí za sebou, ale nejsou vyčítány přes DCMI. Tento způsob řešení nelze použít při snímání sekvence snímků, kdy nám jde o časově velmi přesné prodlevy mezi snímky. Zde je tedy nutné exposure signál vypínat a zapínat znovu až při požadovaném začátku expozice. Díky tomu, že zde mezi snímky uběhne krátká doba, se šum nijak neprojeví.

5.3.2 Odběr jednotlivého snímku

Algoritmus odběru jednoho snímku je velmi triviální, vzhledem k tomu, že se ručně nemusíme starat o žádné časování. Nejprve musíme správně nastavit DCMI a DMA periferie. Zde jde především o nastavení módu a místa kam bude snímek nahrán. Protože je exposure signál ve výchozím stavu, stačí už nyní pouze nastavit capture bit v konfiguračním registru DCMI. Tím je zahájeno čekání na frame valid, po kterém se hotový snímek začne přenášet do SDRAM. Nastaveno je samozřejmě také přerušení informující nás o dokončení přenosu snímku. Vše je přehledně zobrazeno ve vývojovém diagramu 5.5.



Obrázek 5.5. Vývojový diagram odběru jednoho snímku.

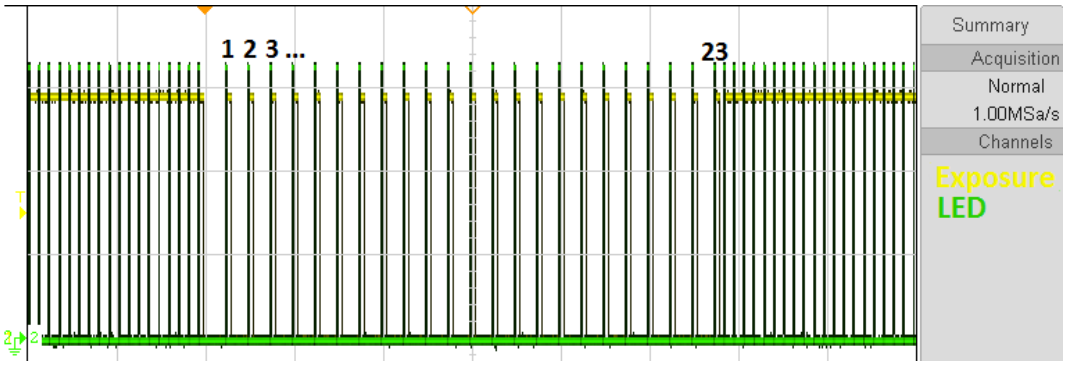
5.3.3 Odběr sekvence snímků

Poněkud komplexnější je již postup pro odběr sekvence snímků. Zde se již musíme starat o časování doby mezi jednotlivými snímky. Před spuštěním snímání je nutné spočítat požadovanou dobu mezi jednotlivými snímky. Než zahájíme expozici prvního snímku je exposure signál přepnut do nuly a čeká se dostatečně dlouhou dobu, aby doběhla expozice předchozího snímku. Tímto způsobem dostaneme senzor do přesně definovaného stavu.

Při prvním průchodu je nejprve nastaveno DCMI a DMA. Následně je nastaven čítač na hodnotu odpovídající době čekání mezi snímky. Nyní je již možně začít expozici prvního snímku nastavením exposure signálu do jedné. Hned potom je spuštěn čítač s nastavenou hodnotou a přepnut capture bit v konfiguračním registru DCMI pro povolení DCMI.

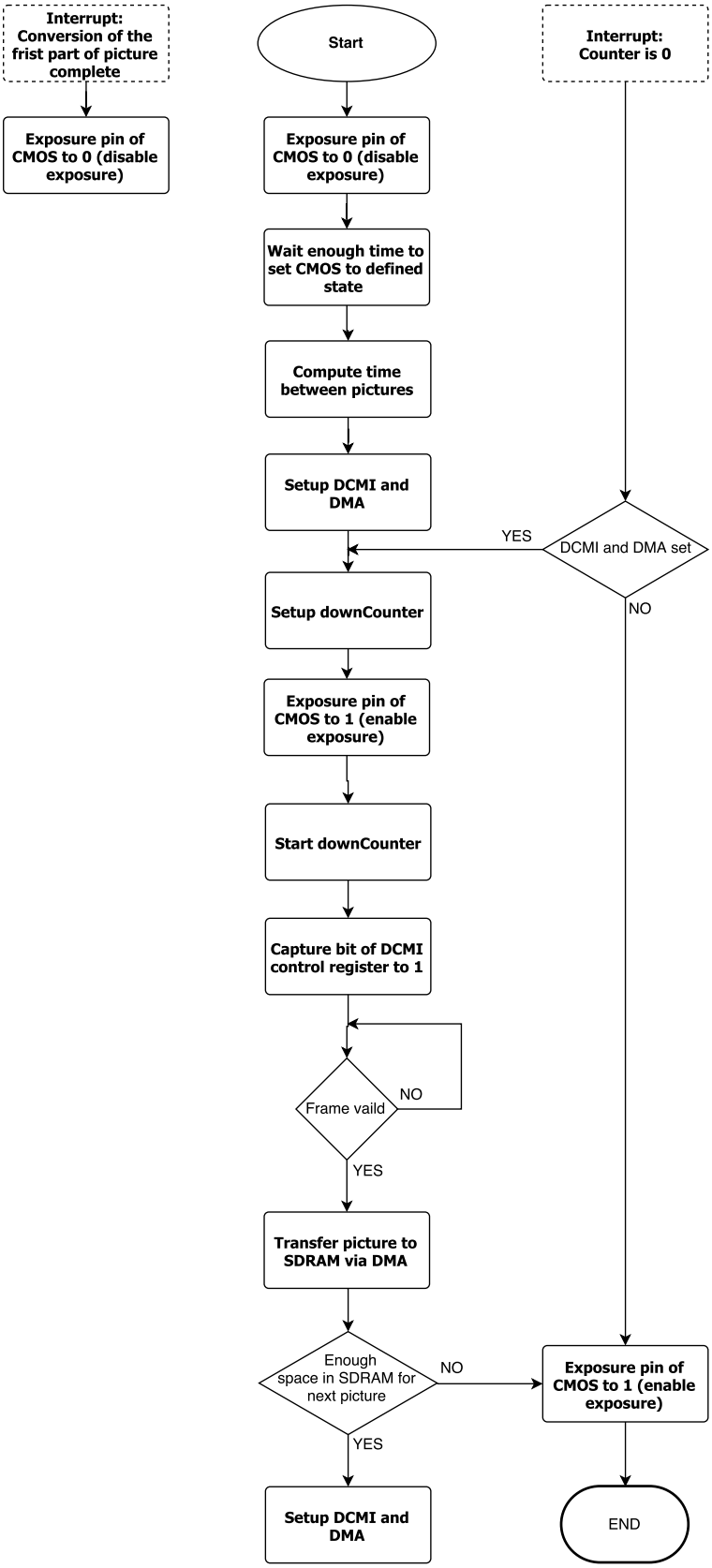
Exposure signál je přepnut zpět do nuly po dokončení přenosu první částí snímku do SDRAM. Po dokončení přenosu celého snímku do SDRAM se znovu nastaví DMA a DCMI tentokrát však s posunutou adresou do SDRAM. Když čítač dočítá do nuly vyvolá se přerušení. V něm je nejprve nastaven exposure singál do jedné, čímž se zahájí expozice (provádí se jako první operace kvůli přesnosti časování). Potom je znovu nastaven a spuštěn čítač. Posledním krokem je nastavení capture bitu konfiguračního registru DCMI. Tento cyklus se opakuje dokud je v SDRAM dostatek místa pro další snímek. Po dokončení odběru je exposure signál nastaven do své klidové hodnoty, která je pro nás v jedné.

Vše je mnohem lépe vidět v následujícím vývojovém diagramu 5.6. Konkrétní časování signálů CMOS senzoru je vyobrazeno na záznamu z osciloskopu 5.7. Žlutý je exposure signál popsaný výše. Zelený je potom signál LED reprezentující přímo dobu expozice. Pokud je v jedné probíhá expozice, pokud v nule neprobíhá expozice. Na záznamu je vidět odběr sekvence snímků plné velikosti. To odpovídá celkem 23 snímkům, jež jsou na obrázku naznačeny. Detail začátku snímání je potom vidět na obrázku 5.8.



Obrázek 5.7. Ukázka časování signálů CMOS senzoru při odběru sekvence snímků.

Obrázek 5.8. Detail časování signálů CMOS senzoru při odběru sekvence snímků.



Obrázek 5.6. Vývojový diagram odběru sekvence snímků.

Kapitola 6
Detekce vibrací

Samotná detekce vibrací probíhá v řídícím počítači. Nejprve je použito hranového detektoru k určení oblasti, ve které budou vibrace sledovány. Následně je třeba získat informaci o nalezených hranách a jejich vhodnou reprezentaci. K tomu je použito základních metod clusteringu. Na takto získané oblasti je konečně možné aplikovat metody hledání malých pohybů a jejich případného zvýrazňování.

6.1 Detekce hran

Hranová detekce se zabývá hledáním ostrých změn v jasové funkci, neboli více formálně hledáním nespojitostí. Oblasti z ostrou změnou jasu obecně odpovídají hranám v obraze. Jejich hledání je netriviální problém, protože v reálném světě těžko zachytíme ideální hranu. Vždy bude obraz zatížen šumy. Poznatky týkající se zpracování obrazu vycházejí z [9].

Pro jednoduchost popisu se prozatím omezíme pouze na 1D případ. Uvažujme tedy 1D spojitou jasovou funkci. Obecně můžeme všechny hranové detektory rozdělit na dvě skupiny, kdy každá používá poněkud jiného principu. První z nich stojí na výpočtu první derivace jasové funkce a následném prostém hledání lokálních maxim. Druhou možností je hledat průchody nulou po aplikaci druhé derivace. Dále se zaměříme pouze na první jmenovanou možnost aplikace první derivace, jelikož právě ta byla implementována.

Nesmíme zapomenout, že v reálném světě má každý obrázek omezenou velikost a konečnou jasovou hloubku. Pohybujeme se tedy v diskrétním světě a derivaci není možné použít. Naštěstí lze velmi jednoduše nahradit derivaci diferencí. Omezíme se tedy na prosté počítání rozdílů jasů jednotlivých pixelů. Počítání pro každý pixel přes celý obrázek by bylo neúnosně výpočetně náročné. Je tedy nutné se ještě omezit na lokální oblast. Díky těmto omezením můžeme výpočet gradientů převést na aplikaci konvoluční masky. Aplikace je potom stejná jako u velkého množství jiných obrazových filtrů.

6.1.1 Diskrétní konvoluce

Konvoluce je při zpracování obrazu používána, protože obraz lze brát pixel po pixelu. V jistém diskrétním čase tedy dávají smysl pojmy jako současný, minulý a následující pixel. Konkrétně v tomto případě nahrazujeme diskrétní čas polohou pixelu. V jednorozměrném případě, kdy máme řádkový obrázek f a řádkovou konvoluční masku g o délce $2M + 1$ můžeme operaci konvoluce zapsat jako:

$$(f * g)[n] = \sum_{m=-M}^{M} f[n - m]g[m] \tag{6.1}$$

Důležitou vlastností konvoluce je komutativita. Platí tedy že:

$$(f * g) = (g * f) \tag{6.2}$$

Aplikace konvoluční masky je ale typicky případ konvoluce dvou dvourozměrných diskrétních funkcí s omezenou velikostí. Uvažujme tedy obrázek A. Dále potom uvažujme konvoluční masku M o velikosti $(2N+1)$x$(2N+1)$, kdy bod $M[0,0]$ leží ve středu masky. Konvoluci takové masky a obrázku potom můžeme přepsat následovně:

$$(A * M)[x,y] = \sum_{i=-N}^{N} \sum_{j=-N}^{N} A[x-i, y-j]M[i,j] \tag{6.3}$$

Celou operaci si potom můžeme snadno představit následovně. Konvoluční maska je tabulka, se kterou jezdíme po obrázku. Střed tabulky je přiložen na konkrétní pixel a následně jsou všechny okolní pixely vynásobeny hodnotou z tabulky, která jej překrývá. Následuje sečtení spočtených hodnoty a jejich přiřazení počítanému pixelu. Tímto způsobem je postupně projet celý obrázek. Například konvoluční maska o velikosti 5x5 se všemi koeficienty 1/25 nebude dělat nic jiného než průměrování.

6.2 Cannyho hranový detektor

Jednou z konkrétních implementací hranových detektorů je Cannyho hranový detektor. Tento algoritmus se skládá hned z několika kroků a tvoří již poměrně komplexní a univerzální detektor. Mezi ostatními hranovými detektory je právě Cannyho považován za jeden z nejspolehlivějších a zároveň tedy i nejpoužívanějších. Obecná kritéria pro hranové detektory zahrnují následující:

- Detekce hran má malou chybovost. Neboli detektor by měl zachytit, co nejvíce existujících hran v obraze.
- Hranové body nalezené operátorem by měly přesně lokalizovat střed hrany.
- Jedna hrana by měla být označena pouze jednou.
- Šum v obraze by neměl být detekován jako hrana.

Je vhodné zmínit, že obrázek, na který byly pro ukázku aplikovány jednotlivé kroky Cannyho detektoru, nepochází z měřící sestavy. Kamera z měřící soustavy má velmi malou hloubku ostrosti a příliš se nehodí na focení běžných objektů. Pomocí mobilního telefonu byl velký objekt zachycen mnohem snáze a čitelněji. Obrázek byl před aplikací jednotlivých kroků upraven pouze snížením rozlišení a převedením do černobílého spektra. Funkce algoritmů také lépe vynikne na takto komplexnějším obrázku, než na obrázku typicky foceném měřící sestavou. Ukázky po aplikaci operátorů pro hledání gradientů jsou invertovány. Většina obrázku je totiž původně černá a po inverzi vše vypadá lépe. Čitelnost se také zlepší a o žádnou informaci nebudeme připraveni.

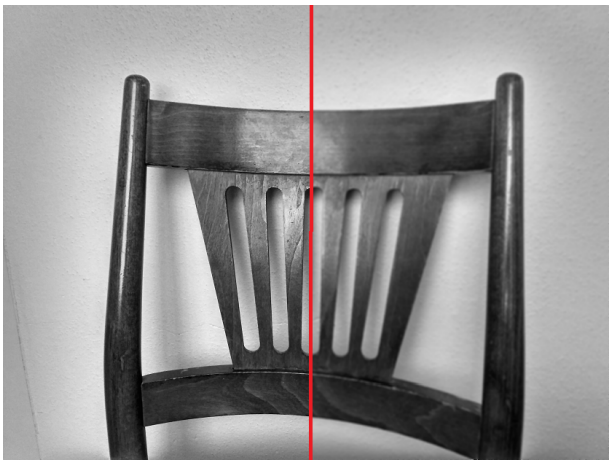
6.2.1 Gaussův filtr

Prvním krokem je aplikace Gaussova filtru, který je jedním ze základních prvků zpracování obrazu. Hranová detekce může být velmi snadno ovlivněna šumem v obraze v podobě falešných hran. Ke snížení vlivu šumu slouží právě Gaussův filtr. Po aplikaci filtru je obraz celkově rozostřen. Tím se připravíme o část detailů, ale zároveň potlačíme vliv šumu.

Použití filtru spočívá v aplikaci vypočítané konvoluční masky na celý obraz. Obecný vzorec pro výpočet prvků konvoluční masky Gaussova filtru o velikosti $(2k+1)$x$(2k+1)$ můžeme zapsat jako:

$$H_{ij} = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right)}; \quad 1 \leq i,\ j \leq (2k+1) \tag{6.4}$$

Je důležité si uvědomit, že velikost konvoluční masky a zvolená velikost σ značně ovlivní chování filtru. Čím větší bude velikost masky, tím více bude obraz rozmazán a hranový detektor bude méně citlivý na šum. To samé platí pro hodnotu σ. Při jejím zvyšování bude rozmazání více znatelné. Obě hodnoty musí být vhodně zvoleny tak, aby byl odstraněn nežádoucí šum, ale zároveň nebyly příliš ovlivněny hledané hrany. Jako ideální se jeví volit velikost masky 3 ($k = 1$) nebo 5 ($k = 2$) a hodnotu sigma v intervalu $\sigma \in \langle 1, 2 \rangle$. Při aplikaci hranového detektoru na obraz s velmi jednoduchou strukturou ovšem není nutné Gaussův filtr vůbec používat. Ukázku aplikace Gaussova filtru můžeme vidět na obrázku 6.1.



Obrázek 6.1. Vlevo původní část obrázku a vpravo po aplikaci Gaussova filtru.

6.2.2 Hledání gradientů

Po aplikaci Gaussova filtru přichází ten nejdůležitější krok každého hranového detektoru. Je nutné spočítat konkrétní hodnoty gradientů v obraze. Cannyho hranový detektor používá výpočet velikosti gradientu aproximovaný pomocí operátoru. Celý výpočet spočívá v aplikaci konvoluční masky na celý obrázek. Po aplikaci vhodné konvoluční masky získáme velikosti gradientů v daném směru.

Ve většině případů si vystačíme s výpočtem gradientu v horizontálním a vertikálním směru. Zvláště v případě kdy předpokládáme hrany především v těchto dvou směrech. Při výpočtu gradientů ve více směrech sice získáváme více informací, ale každý směr znamená aplikaci konvoluční masky na celý obraz a tedy i značné zvyšování výpočetní náročnosti, která už tak není nízká. Pro obecnou symetrickou konvoluční masku M a zpracovávaný obraz A potom můžeme napsat výpočet ve zmíněných směrech jako:

$$G_x = M * A \; ; \; G_y = M^T * A \tag{6.5}$$

Kde G_x je velikost gradientu v horizontálním směru a G_y je velikost gradientu ve vertikálním směru.Celkovou velikost gradientu G potom jednoduše vypočteme jakožto:

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{6.6}$$

Ze známých směrových gradientů jsme schopni spočítat i směr gradientu Φ a to ze vzorce:

$$\Phi = \mathrm{atan2}(\mathrm{G_x}, \mathrm{G_y}) \tag{6.7}$$

V práci byly implementovány hned 4 různé operátory reprezentující vybrané konvoluční masky. Každý z těchto operátorů má trochu jiné vlastnosti a hodí se pro detekci různých

hran. Srovnání vybraných operátorů je vyobrazeno na obrázku 6.2. Zvoleny byly pouze symetrické operátory o velikosti 3x3 vhodné na aplikaci pouze ve dvou směrech. Ještě můžeme zmínit, že jsou všechny tyto operátory separabilní a je tedy možné při případné implementaci ve vývojovém kitu optimalizovat výpočet. Implementovány byly následující operátory:

- Prewitt
  Jeden ze základních operátorů dávající stejnou váhou přímému i úhlopříčnému směru.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} * A \; ; \; G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} * A \tag{6.8}$$

- Sobel
  Operátor dávající větší váhu přímému směru. Velmi oblíbený a často používaný operátor.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \; ; \; G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A \tag{6.9}$$

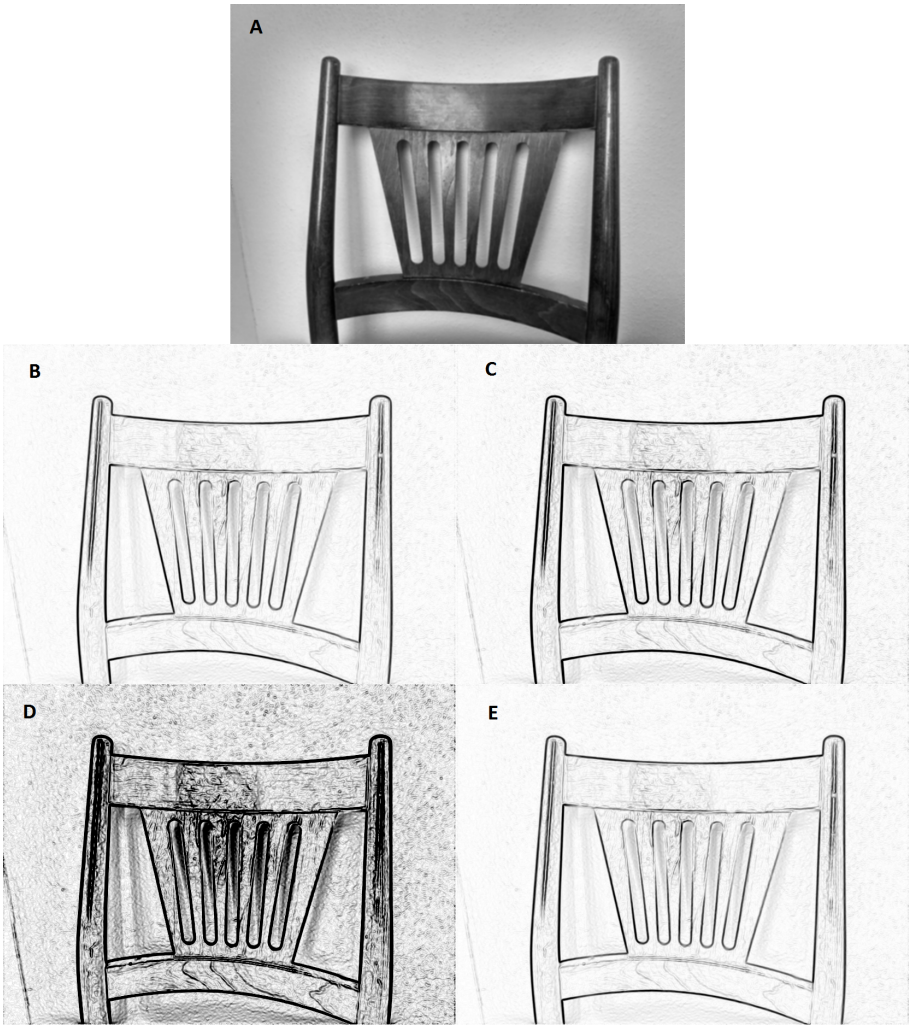
- Scharr
  Rozložením středu a krajů je velmi podobný Sobelu operátoru, zároveň je ale velmi citlivý.

$$G_x = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} * A \; ; \; G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} * A \tag{6.10}$$

- Robinson
  Jediný implementovaný operátor dávající váhu i středu. Vhodný spíše na vícesměrové aplikace, ale použitelný i na dva směry.

$$G_x = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix} * A \; ; \; G_y = \begin{bmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix} * A \tag{6.11}$$

Existuje velké množství dalších operátorů používaných na hranou detekci. Značná část z nich je nesymetrická a tak byly rovnou vyřazeny. Pravděpodobně dva nejznámější neimplementované operátory jsou Robertsův kříž a Laplaceův operátor. První jmenovaný nebyl implementován jednak kvůli jeho asymetrii a především kvůli vysoké citlivosti na šum. Laplaceův operátor potom na rozdíl od použitých operátorů aproximuje druhou derivaci. Používá tedy poněkud jiný přístup a má i výrazně jiné vlastnosti. Jeho konvoluční masku stačí aplikovat jednou a je tedy méně výpočetně náročný. Na druhou stranu má mnoho nevýhod. Po aplikaci ztratíme informaci o směru gradientu (hrany), je velmi citlivý na šum a často zdvojuje hrany. Proto bylo rozhodnuto ho vůbec neimplementovat.



Obrázek 6.2. Porovnání operátorů aplikovaných na obrázek A (aplikovaný Gaussův filtr) (invertováno). A.) Původní B.) Prewitt C.) Sobel D.) Scharr E.) Robinson

6.2.3 Non-maximum potlačení

V dalším kroku je možné aplikovat techniku ztenčování hran zvanou Non-maximum potlačení. Tento algoritmus pomáhá potlačit hodnoty gradientů, které nejsou rovny lokálnímu maximu. Je tedy jasné, že k aplikaci již potřebujeme napočítané hodnoty gradientů i jejich směry.Algoritmus se aplikuje na každý pixel obrázku ve dvou krocích:

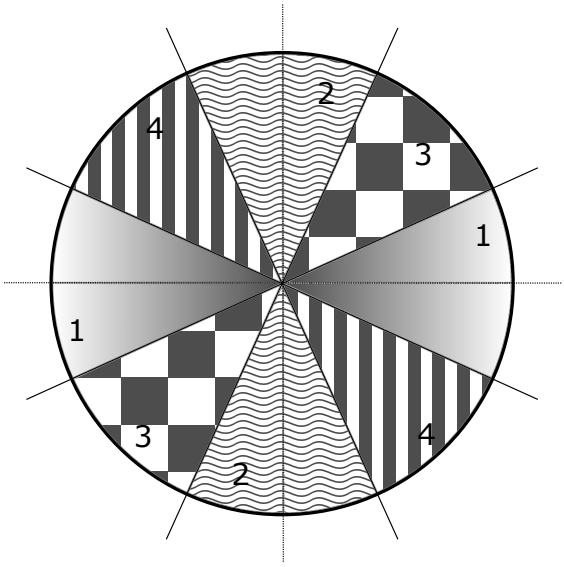
1. Porovnání velikostí gradientu v kladném i záporném směru gradientu.
2. Pokud je velikost gradientu nejvyšší v daném směru a oblasti, je hodnota zachována. Jinak je potlačena.

Stejně jako ve většině případů se omezujeme na malou oblast okolo vybraného pixelu a používáme v podstatě aplikaci konvoluční masky o velikosti 3x3. Jako okolí pixelu v daném směru tedy vždy uvažujeme pouze dva okolní pixely. Směry gradientu jsou tedy ještě před aplikací Non-maximum potlačení zařazeny do jednoho ze 4 kvadrantů 6.3.

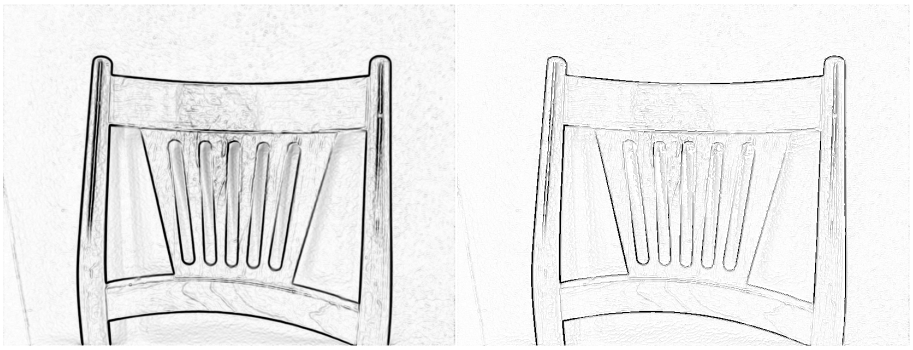
Je vhodné ještě zmínit, že znaménko směru gradientu je pro nás nedůležité. Uvažujeme totiž pouze velikosti gradientů a díky tomu stačí počítat pouze se čtyřmi směry. Ukázku aplikace Non-maximum potlačení můžeme vidět na obrázku 6.4

Přestože byla tato část hranové detektoru funkčně implementována, není ji ve většině případů vhodné použít. Cílem celé naší hranové detekce je získat konkrétní oblast,

ve které se budeme zabývat detekcí vibrací. Aplikace Non-maximum potlačení nám sice ztenčí hrany, ale zároveň výsledky hranové detekce může zkreslit. Při vhodně nastavených úrovních pro prahování jej není nutné použít. Samozřejmě je možné narazit na situace, kdy bude aplikace potlačení přínosná, a proto bylo zachováno jako volitelná možnost.



Obrázek 6.3. Nákres kvadrantů směru gradientu.



Obrázek 6.4. Ukázka Non-Maximum potlačení. Vlevo obrázek před aplikováním non-Max, vpravo po aplikování (invertováno).

6.2.4 Prahování

Po spočtení gradientů a možné aplikaci Non-maximum potlačení zůstává ve většině případu v obraze stále poměrně hodně šumu. K jeho potlačení je používáno prahování. Prahování je jednou ze základních a nejjednodušších metod zpracování obrazu. Je vhodně vybrán práh a následně jsou všechny hodnoty menší než hodnota prahu potlačeny.

V oblasti hranové detekce se ve většině případů využívá dvou prahů. Pokud je velikost gradientu vyšší než horní práh je pixel označen jako silná hrana. Pokud je velikost gradientu vyšší než dolní práh je pixel označen jako slabá hrana. Všechny zbylé pixely s velikostí gradientu nižší než dolní práh jsou potlačeny. Implementace prahování je tedy velmi jednoduchá. Problematická je volba prahů. Jejich ideální určení ve všech případech je netriviální úloha, protože jsou ovlivněny mnoha faktory.

Byla zachována možnost uživatelského nastavení prahů, ale je vhodné pokusit se i o jejich automatické nastavení. Rozhodl jsem se zkusit implementovat Otsuho metodu prahování pro určení horního prahu. Dolní práh je pak určen jako 1/2 horního prahu, což odpovídá typickému poměru horního a dolního prahu 2:1 (někdy používáno 3:1 nebo 3:2).

6.2.5 Otsuho metoda prahování

Otsuho metoda prahování je používána nejčastěji na oddělení pozadí od popředí a počítá tedy s jasovými hodnotami pixelů. Je ji však možné použít i na hodnoty gradientů, což je využito v našem případě. Cílem algoritmu je rozdělit obraz do dvou tříd, tak aby jejich intra-class rozptyl byl minimální.

Nejprve je nutné si zavést několik statistických veličin pro diskrétní obory se stejně pravděpodobnými hodnotami:

1. Střední hodnota

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6.12}$$

2. Rozptyl

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^2 \tag{6.13}$$

Kde x_i jsou prvky oboru a n je počet prvků v oboru. Intra-class rozptyl ${\sigma_W}^2$ potom můžeme zadefinovat jako:

$${\sigma_W}^2(t) = w_0(t){\sigma_0(t)}^2 + w_1(t){\sigma_1(t)}^2 \tag{6.14}$$

Kde w_0 a w_1 jsou pravděpodobnosti tříd rozdělených prahem t a σ_0 a σ_1 jsou rozptyly daných tříd. Pravděpodobnosti tříd můžeme také v našem případě nazývat váhou. Protože cílem je rozdělit obraz na pozadí a popředí budeme používat indexy b pro pozadí a f pro popředí. Váhu neboli pravděpodobnost potom můžeme vyjádřit jako:

$$W_b(t) = w_0(t) = \sum_{i=0}^{t-1} p(i) = \frac{1}{n}\sum_{i=0}^{t-1} s(i) \tag{6.15}$$

$$W_f(t) = w_1(t) = \sum_{i=t}^{n} p(i) = \frac{1}{n}\sum_{i=t}^{n} s(i) \tag{6.16}$$

Kde $p(i)$ je pravděpodobnost výskytu hodnoty i, $s(i)$ je počet výskytů hodnoty i, n je celkový počet prvků a t je práh rozdělující třídy.

Počítaný intra-class rozptyl ${\sigma_W}^2$ můžeme ještě jednou přepsat následovně:

$${\sigma_W}^2(t) = W_b(t){\sigma_b}^2(t) + W_f(t){\sigma_f}^2(t) \tag{6.17}$$

Počítání těchto hodnot je ale výpočetně poměrně náročné. Naštěstí je dokázáno, že hledání minima intra-class rozptylu je ekvivalentní hledání maxima inter-class rozptylu. Jeho výpočet je mnohem jednodušší a tedy i hledání jeho maxima nezabere tolik výpočetního času. Inter-class rozptyl ${\sigma_B}^2$ je zadefinován:

$$\begin{aligned} {\sigma_B}^2(t) = \sigma^2 - {\sigma_W}^2(t) = W_b(t)\left[\mu_b(t) - \mu\right]^2 + W_f(t)\left[\mu_f(t) - \mu\right]^2 = \\ = W_b(t)W_f(t)\left[\mu_b(t) - \mu_f(t)\right]^2 \end{aligned} \tag{6.18}$$

Cílem je tedy najít takový práh t, pro který je hodnota inter-class rozptylu maximální. Při zapnutí možnosti automatického počítání prahů je tato hodnota počítaná pro každý obrázek zvlášť. Výsledný algoritmus v pseudokódu bude vypadat následovně:

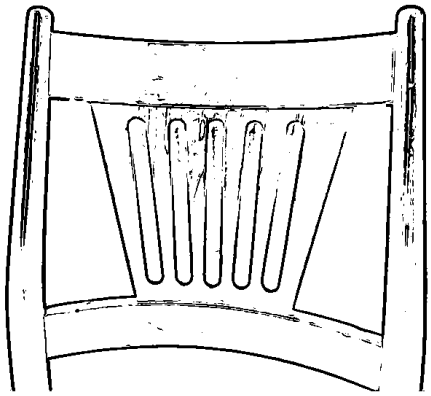
```
Wb; // background weight
Wf; // foreground weight
ub; // background mean
uf; // foreground mean
varBetween; // between variance (inter-class)
maxVar; // maximum between variance (inter-class)
t; // chosen threshold

computeHistogram();

maxVar = 0;
t=0;
for(i=0;i<HistogramLength;i++){
    update Wb;
    update Wf;
    update ub;
    update uf;
    compute varBetween;
    if(maxVar<varBetween){
       maxVar=varBerween;
       t=i;
    }
}
```

6.2.6 Hysterezní sledování hran

Po aplikaci prahování máme v obraze pixely pouze ve třech stavech. Z toho jsou zajímavé dva a to silné a slabé hrany. Posledním krokem algoritmu je rozhodnutí, které slabé hrany zachovat a které potlačit. Je možné potlačit všechny slabé hrany nebo naopak ze všech slabých hran udělat silné hrany. Zajímavější je ale takzvané hysterezní sledování hran. To říká, že pokud je pixel slabé hrany přímo spojen se silnou hranou, bude zachován. Naopak pokud není přímo spojen se silnou hranou bude potlačen. Po aplikaci tohoto kroku již máme finální obrázek s binární informací, kde se hrany nachází (ukázka na obrázku 6.5).



Obrázek 6.5. Výsledek hranové detekce (invertováno).

6.3 Clustering hran

Pro další zpracování dat je vhodné detekované hrany reprezentovat i jinak než jen bílými pixely v obraze. Cílem je získat nejprve vhodnou reprezentaci jednotlivých hran. Následně odfiltrovat hranové pixely nenáležící žádné hraně způsobené šumem. Takto získané hrany k sobě musíme přiřadit mezi jednotlivými obrázky ze sekvence. Po namapování stejných hran přes všechny obrázky již můžeme určit oblast, ve které budou vyhodnocovány vibrace.

6.3.1 Prohledávání stavového prostoru hran

Nejjednodušším způsobem pro získání reprezentace hran je použít prostého rekurzivního prohledávání stavového prostoru přes pixely s hodnotou 255. Při větším počtu hranových pixelů tvořících jednu hranu však velmi snadno může dojít k přetečení zásobníku. Nabízí se přepsaní algoritmu do iterativní podoby s použitím vlastního zásobníku a objektu na předávání parametrů. Tímto způsobem je zamezeno přetečení zásobníku. Nejlépe se výsledný algoritmus ukáže na následujícím pseudokódu:

```
foreach(pixel in picture){
    if(pixel == 255){
        new edge;
        stack.push(pixel);
        edge.add(pixel);
        pixel = 0;
        while(!stack.isEmpty()){
            pixel=stack.pop();
            foreach(pix in pixel.neighbors){
                if(pix == 255){
                    stack.push(pix);
                    edge.add(pix);
                    pix = 0;
                }
            }
        }
    }
}
```

Protože ve většině případů následně procházíme všechny pixely celé hrany, je možné pro jejich reprezentaci použít spojového seznamu. Není třeba ukládat hodnoty do stromové struktury, protože přínos by byl minimální.

Tímto způsobem získáme jednotlivé hrany reprezentované pomocí spojových seznamů. Jako hrana se však jeví i velmi malé skupiny hranových pixelů způsobených šumem a nepřesnou hranovou detekcí. Pro odfiltrování jsou hrany nejprve seřazeny podle velikosti (počtu pixelů v hraně). Následně zachováváme pouze hrany, které nejsou více než pětkrát menší než předchozí hrana a zároveň byly zachovány veškeré předchozí hrany. Tento způsob se může zdát velmi primitivní, ale ukázal se jako poměrně spolehlivý. Výhodné je, že stačí najít první hranu nevyhovující kritériu. Všechny menší hrany jsou považovány za šum a tedy potlačeny.

6.3.2 Určení oblasti detekce vibrací

Dalším důležitým krokem pro získání reprezentace hran. je jejich namapování přes jednotlivé obrázky sekvence. Pro následující zpracování je nezbytné správně přiřadit stejné hrany v sekvenci k sobě. Před samotným přiřazení je však potřeba provést ještě jeden krok. Kvůli nepřesnosti hranové detekce se někdy může stát, že je hrana v některých obrázcích rozdělena do dvou částí (části nejsou spojeny hranovým pixelem). Tento případ je třeba detekovat a hranu spojit. Následně jsou již hrany mezi obrázky přiřazovány k sobě pomocí minimalizace podobnostní funkce hran. Ta je definováno jako:

$$S(E_1, E_2) = |h_1 - h_2| + |w_1 - w_2| + |s_1 - s_2| + TL(E_1, E_2) + BR(E_1, E_2) \tag{6.19}$$

Kde E_1 a E_2 jsou porovnávané hrany. Hodnota h [px] je výška hrany a w [px] šířka hrany, které jsou definovány jako výška a šířka obdélníku opsaného hraně. Velikost hrany je značena s a je definována jako počet pixelů náležící hraně. Funkce $TL(E_1, E_2)$ a $BR(E_1, E_2)$ jsou potom eukleidovské vzdálenosti horních levých pixelů (TL) porovnávaných hran a dolních pravých pixelů (BR).

Takto zapsaná podobnostní funkce dostatečně dobře reprezentuje vlastnosti hrany a zároveň je stále velmi jednoduchá. Pro naše účely se ukázala jako dostačující a tudíž správně fungující. Snímaný prostor ve většině případů nebude členitý, a proto je takováto funkce dostačující.

Po namapování hran v sekvenci už zbývá pouze určit oblast detekce vibrací. Uživatel nejprve vybere sledovanou hranu. Následně je oblast určena jakožto sjednocení množin pixelů všech k sobě přiřazených hran mezi snímky sekvence. Přidány jsou také pixely ležící „mezi“ vybranými pixely, tak aby byla oblast celistvá. Oblast je nakonec ještě zvětšena způsobem, aby se zamezilo možným nepřesnostem hranové detekce a oblast vždy skutečně obsahovala celou sledovanou hranu.

6.4 Detekce vibrací ve vybrané oblasti detekce

Jak již bylo několikrát zmíněno, hlavním cílem celé této práce je ověřit možnosti detekce vibrací pomocí informace z obrazového senzoru. Díky všem předchozím krokům máme vhodně nasnímaná data a určenou oblast, ve které se vybraná hrana vyskytuje v celé sekvenci snímků. Nyní zbývá poslední krok, a to určení zda se v této oblasti vyskytují hledané vibrace.

Protože předpokládáme hledání velmi malých pohybových změn (menších než jeden pixel), není možné hledat změny přímo na detekovaných hranách. To je možné pouze pro větší pohyby, kde by se výrazně měnila poloha detekované hrany. Pro hledání pohybových

změn menších než jeden pixel je to metoda spíše nepoužitelná, protože hranová detekce je velmi velký zásah do původních dat a může je výrazně zkreslit. Hrozila by potom například falešná kladná detekce nebo naopak falešná záporná detekce. Abychom nezpracovávali zkreslená data, musíme se zaměřit na původní snímky. Jako možnost hledání změn menších než jeden pixel se potom nabízí zaměření se na jasové změny pixelů. Nápad využití změny jasových hodnot pixelů pro detekci pohybu vychází z [10].

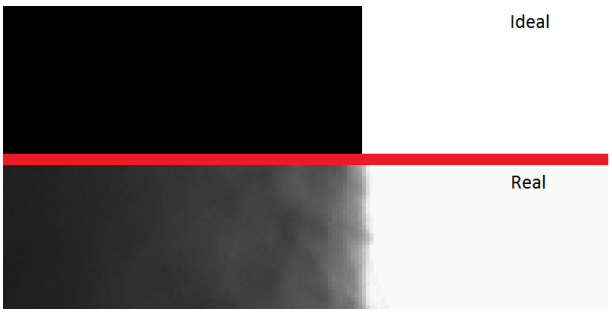
6.4.1 Výpočet jasových změn mezi obrázky

Jasové změny mezi obrázky jsou počítány pouze přes oblast vybranou v předchozích krocích. Nejprve jsou vypočítány prosté rozdíly mezi jednotlivými pixely na stejných pozicích v po sobě jdoucích snímcích. Matematicky můžeme tento vztah obecně zapsat jako:

$$\Delta b(n,(x,y)) = b(n,(x,y)) - b(n+1,(x,y)) \; ; \quad n \in \langle 1, N-1 \rangle \; ; \; (x,y) \in A \tag{6.20}$$

Kde $\Delta b(n,(x,y))$ je počítaná jasová změna pixelů na pozici (x,y) mezi snímky n a $n+1$. Jasová hodnota pixelu na pozici (x,y) ve snímku n je $b((n,(x,y))$ a A je množina všech pixelů ve vybrané oblasti detekce. Počet snímků v sekvenci je potom označen jako N. Hodnoty $\Delta b(n,(x,y))$ jsou spočteny pro všechna platná n (rozdíly mezi všemi obrázky) a pro všechna (x,y) náležící do A (všechny pixely ve vybrané oblasti).

Detekovat chceme vždy pohyb pouze v jednom směru současně. Při zaměření se na hranu je to pro nás směr kolmý k hraně. Ve směru rovnoběžném s hranou by to ani nebylo možné. Pro další zpracování je nutné správně určit orientaci vybrané oblasti detekce. Ta ovšem nemusí být nutně obdélníková. Spočteme tedy přes kolik řádků obrázku se rozkládá a maximální počet sloupců kolik zabírá na řádku. Větší z hodnot prohlásíme za výšku (H) a menší za šířku (W_y). Výška oblasti by měla být vždy konstantní. Na druhou stranu šířka se může měnit v závislosti na poloze v ose y (výšce). Na takto zadefinovanou oblast již můžeme obecně aplikovat další kroky.



Obrázek 6.6. Porovnání ideální a reálné hrany.

Jelikož hrana v obrázku není zdaleka ideální, je při jejím pohybu jasová změna rozprostřena na více pixelů v daném řádku. Srovnání ideální a reálné hrany je vidět na obrázku 6.6.

Jasová změna je tedy sečtena přes jednotlivé řádky oblasti detekce následujícím způsobem:

$$\Delta b(n, y) = \sum_{x=1}^{W_y} \Delta b(n, (x, y)) \tag{6.21}$$

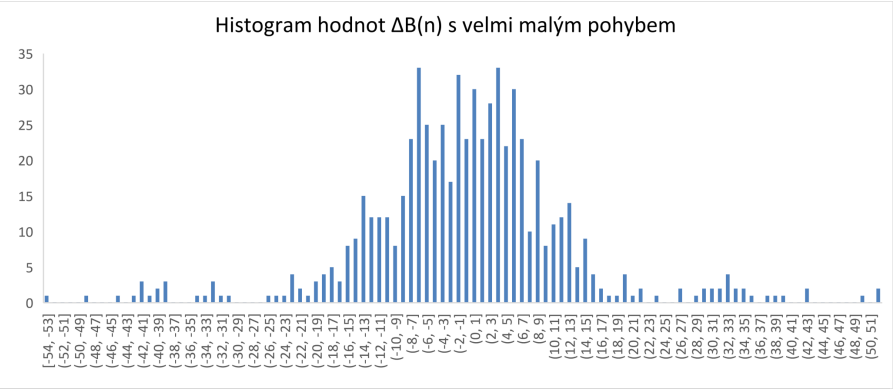
Kde W_y [px] je šířka daného řádku y (počet pixelů v řádku) a $\Delta b(n, y)$ je celková jasová změna pro řádek y a změnu n. Tyto sumy jsou provedeny opět pro všechna platná n a zároveň přes všechny řádky y ve vybrané oblasti detekce A.

Spočtené hodnoty jasových změn pro jednotlivé řádky $\Delta b(n, y)$ se kvůli šumům mohou mírně lišit i v rámci jedné počítané změny n. Pro získání jednotné reprezentace jasové změny přes řádky mezi obrázky je tedy proveden aritmetický průměr hodnot $\Delta b(n, y)$ pro změnu n následovně:

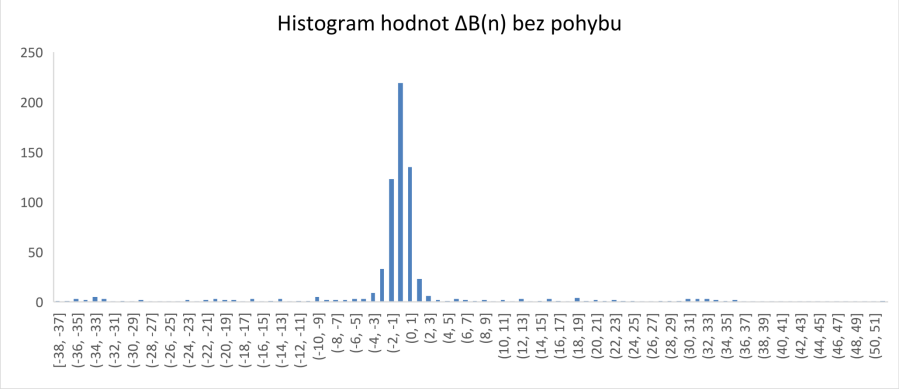
$$\Delta B(n) = \frac{1}{H} \sum_{y=1}^{H} \Delta b(n, y) \tag{6.22}$$

Kde H [px] je výška oblasti detekce A a $\Delta B(n)$ je průměrná jasová změna přes řádky v oblasti detekce A pro změnu n.

Hodnoty $\Delta B(n)$ jsou již informace, které hledáme. Je tedy vhodné před dalším zpracováním určit, zda se zde pohyb vůbec nachází a má tedy smysl pokračovat. I při snímání statického objektu budou vždy mezi obrázky určité jasové změny způsobené šumem. Většina těchto změn ale bude natolik malá, že jsme schopni určit zda se objekt nehýbe, nebo zda je pohyb objektu menší než rozlišovací schopnost detekce. Cílem je zamezení dalšího zpracování nesmyslných dat, které by i tak jistě nepřinesly správné výsledky. Data získána ze snímků nepohybujícího se objektu (nebo pohybujícího se pod rozlišovací naší rozlišovací schopnost) se zásadně liší od dat získaných ze snímků pohybujícího se objektu. Data jsou rozlišena pomocí histogramu jasových změn $\Delta B(n)$. Pokud se více než 50% hodnot histogramu vyskytuje v intervalu $\langle -2, 2 \rangle$, není nutné data dále zpracovávat a je prohlášeno, že pohyb nebyl nalezen. Ukázka histogramů pro oba případy je na zobrazena na obrázcích 6.7 a 6.8.



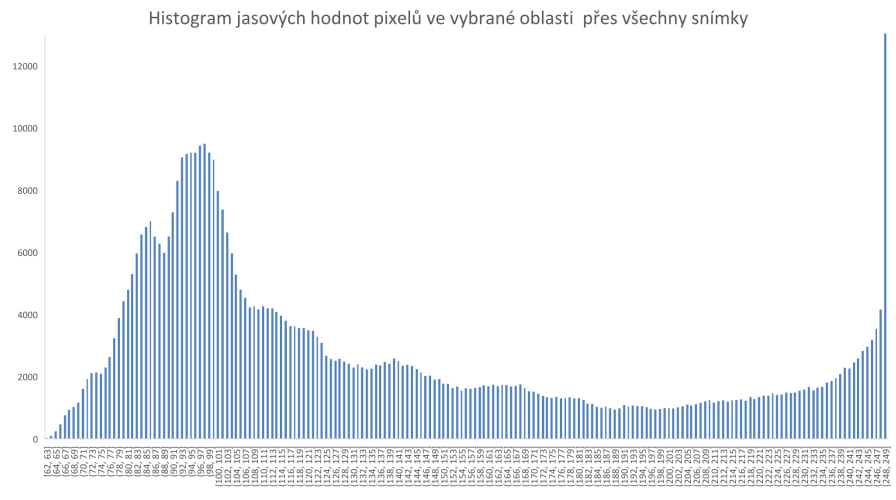
Obrázek 6.7. Histogram hodnot $\Delta B(n)$ v případě pohybu těsně nad rozlišovací schopnost.



Obrázek 6.8. Histogram hodnot $\Delta B(n)$ bez pohybu.

6.4.2 Určení jasové změny odpovídající pohybu

Po předchozím kroku již máme pohyb v oblasti reprezentovaný pomocí jasových změn přes řádky mezi obrázky. Zatím však nemáme žádné měřítko, pomocí kterého bychom mohli tyto změny přepočítat na změny polohy. Je tedy nutné nějakým způsobem určit jak velká jasová změna přes řádek odpovídá pohybu o jeden pixel. V případě ideální hrany je to velmi jednoduché. Je to prostý rozdíl mezi světlou částí před hranou (pozadí) a tmavou částí za hranou (objekt). Hrana totiž není tvořena jasovým gradientem a je tedy možné změnu spočítat z libovolných bodů objektu a pozadí z dané oblasti. V případě ideální hrany by i změna přes řádek byla koncentrována pouze v tolika pixelech, o kolik proběhl pohyb.



Obrázek 6.9. Histogram jasových hodnot pixelů ve vybrané oblasti detekce přes všechny snímky v sekvenci.

Reálná hrana je ovšem tvořena jasovým gradientem a takto jednoduchý postup nelze použít. Jelikož se ale jedná o analogický problém, můžeme z něj vyjít. Stále to bude rozdíl hodnot reprezentujících oblast před hranou (pozadí) a za hranou (objekt). Řešení jsem se pokusil najít znovu v histogramu. Ten je spočítán pro jasové hodnoty pixelů ve vybrané oblasti detekce pro všechny snímky ze sekvence najednou.

V histogramu předpokládáme dvě hlavní lokální maxima. Jeho podoba na reálných datech je vidět na obrázku 6.9 (histogram je oříznutý na počet 12000, nejvyšší jasová

hodnota má ve skutečnosti více než 10x tolik výskytů). Jedno bude posunuté v histogramu doleva L_{max} a bude odpovídat tmavé části oblasti (objektu). Druhé bude posunuto naopak doprava R_{max} a bude odpovídat pozadí. Tyto maxima jsou nalezena. Následně je spočítán jejich rozdíl, který prohlásíme za jasovou změnu odpovídající změně o jeden pixel B_c.

$$B_c = R_{max} - L_{max} \tag{6.23}$$

6.4.3 Přepočet do reálných jednotek vzdálenosti

S dosud dostupnými informace jsme schopni vypočítat změnu polohy mezi obrázky v pixelech $\Delta P(n)$ [px] dle následujícího vztahu:

$$\Delta P(n) = \frac{\Delta B(n)}{B_c} \tag{6.24}$$

Prakticky by ale bylo příhodné tyto hodnoty přepočítat do jednotek vzdálenosti. K tomu je nutné znát, z jaké vzdálenosti byl objekt snímán a také parametry objektivu a senzoru. Dále předpokládáme, že na hranu bylo zaostřeno. Bez toho by v podstatě nebylo možné vypočítat hodnotu na přepočet do jednotek vzdálenosti. Díky předchozímu předpokladu můžeme ze vzdálenosti předmětu od čočky a spočítat vzdálenost vzdálenost čočky od senzoru. Protože předpokládáme zaostření na vzdálenost ve které se nacházel předmět (hrana), je možné uvažovat vzdálenost senzoru od čočky rovnou vzdálenosti obrazu předmětu od čočky a'. Pro následující výpočet si tedy znovu vystačíme se zobrazovací rovnicí pro tenkou čočku [2]. Naším cílem je výpočet zvětšení objektivu. Tentokrát nejprve použijeme Gaussovu zobrazovací rovnici ve tvaru:

$$\frac{1}{f} = \frac{1}{a} + \frac{1}{a'} \tag{6.25}$$

Kde f [mm] je ohnisková vzdálenost, a [mm] je vzdálenost předmětu od čočky a a' [mm] je vzdálenost obrazu předmětu (vzdálenost senzoru) od čočky. Ze vztahu (6.25) potom pouze vyjádříme vzdálenost senzoru od čočky a' a můžeme vypočítat hledanou hodnotu:

$$a' = \frac{f \cdot a}{a - f} \tag{6.26}$$

Když známe vzdálenost senzoru od čočky, můžeme již velmi snadno spočítat zvětšení objektivu β. Nejprve si vyjádříme hledané zvětšení přes velikost předmětu y [mm] a velikost obrazu předmětu y' [mm]:

$$\beta = \frac{y'}{y} \tag{6.27}$$

Vztah mezi y, y', a, a' odvodíme z již jednou použitého obrázku obrázku 3.2:

$$\tan\alpha = \frac{y}{a} = \frac{y'}{a'} \Rightarrow \frac{y'}{y} = \frac{a'}{a} \tag{6.28}$$

Můžeme tedy psát že zvětšení β se rovná:

$$\beta = \frac{y'}{y} = \frac{a'}{a} \tag{6.29}$$

Pokud bychom se chtěli vyhnout výpočtu hodnoty a' můžeme zvětšení vyjádřit i pomocí vzdálenosti předmětu od předmětového ohniska z [mm] a ohniskové vzdálenosti f [mm]:

$$\beta = \frac{a'}{a} = \frac{f + z'}{f + z} = \frac{f + \frac{f^2}{z}}{f + z} = \frac{f\left(1 + \frac{f}{z}\right)}{f + z} = \frac{f\left(\frac{z+f}{z}\right)}{f + z} = \frac{f}{z} \tag{6.30}$$

Když už víme velikost zvětšení, stačí pouze zjistit velikost užitečné plochy senzoru. Při známe velikosti užitečné plochy senzoru a jeho rozlišení snadno dopočítáme reálnou velikost odpovídající jednomu pixelu na senzoru. Pro senzor použitý v této práci 3.2 vychází velikost jednoho pixelu přibližně 6x6 μm. Při uvažování pohybu ve vertikálním nebo horizontálním směru potom spočteme, že jeden pixel odpovídá reálně velikosti:

$$l = \frac{y'}{\beta} = \frac{6}{\beta} \tag{6.31}$$

Kde l [μm] je tedy velikost objektu promítaného přesně na jeden pixel. Po spočtení této hodnoty již snadno můžeme přepočítat hodnoty změn z pixelů na μm.

6.4.4 Přepočet změn polohy na absolutní polohu

Abychom mohli spočítat amplitudu pohybových vibrací, musíme znát hodnoty absolutní polohy v jednotlivých snímcích. Prozatím známe pouze změny polohy mezi dvěma po sobě jdoucími obrázky. Pro začátek zvolíme počáteční polohu v nule a prostě postupně budeme sčítat známe změny polohy. Tímto způsob již získáme požadovaný průběh.

Jelikož ale nemůžeme vědět z jaké polohy pochází první snímek, byl původní předpoklad vložit počátek do nuly nesprávný. Je totiž více než pravděpodobné, že počáteční poloha nebyla nulová. Celý spočítaný průběh polohy hrany v čase tedy může být posunutý právě o počáteční polohu. Abychom mohli správně určit amplitudu, musíme se pokusit toto posunutí spočítat a celý průběh následně posunout.

Posun můžeme zkusit odhadnout například následujícím postupem. Nejprve vybereme 10% nejvyšších a 10% nejnižších bodů polohy. Následně provedeme aritmetické průměry těchto dvou oborů hodnot. Nazvat je můžeme například Max_m a Min_m. Posun z těchto hodnot potom odhadneme jako:

$$offset = \frac{Max_m + Min_m}{2} \tag{6.32}$$

Což je vlastně průměr těchto dvou hodnot.V případě předpokládaného sinusového průběhu by měl být tento průměr roven nule. Pokud má jinou hodnotu, je tato hodnota přímo hledaný offset.

Protože předpokládáme sinusový průběh pohybu, nabízela by se možnost udělat průměr přes všechny hodnoty polohy. Při dostatečném počtu period vychází průměr sinu velmi blízký nule i při necelém počtu period. My se však snažíme určit posun průběhu správně v co nejvíce případech. Ve většině případů budeme vzorkovat sinusový pohyb výrazně pomaleji než je jeho frekvence. Nesplníme tedy podmínku vzorkovacího teorému a může docházet k aliasingu (viz. 5.2.1). Může tedy nastat případ, kdy většina vzorků bude kladná nebo záporná. Průměr přes všechny hodnoty by byl takto velmi snadno rozvážen a posun by byl určen špatně. Když vezmeme určité procento okrajových hodnot odhad by měl být úspěšný ve více případech.

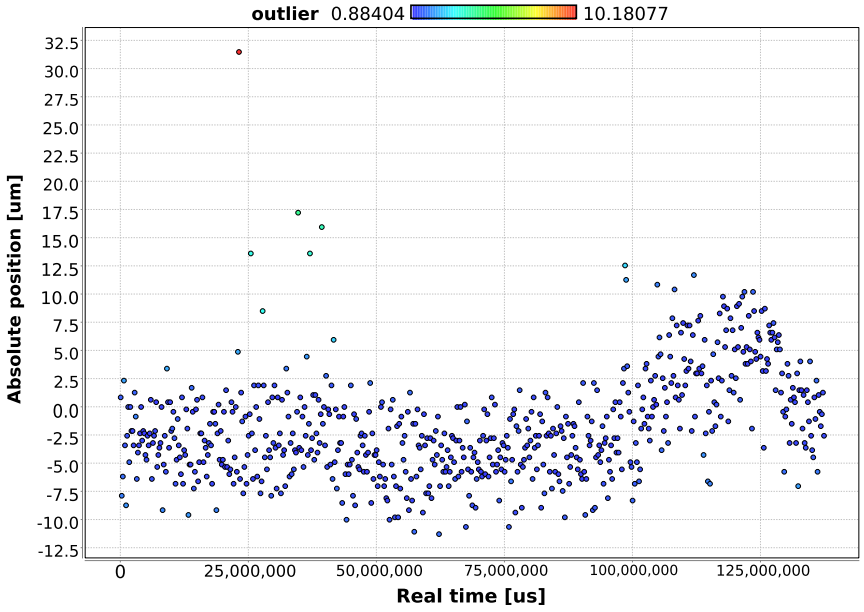
Po určení posunu průběhu už pouze zbývá všechny hodnoty o zjištěný offset posunout a časový průběh polohy je připraven na výpočet amplitudy.

6.4.5 Výpočet amplitudy vibrací

Posledním krokem celé detekce vibrací je určení jejich pro nás nejdůležitějšího parametru, kterým je amplituda. Abychom odstranili výrazné extrémy způsobené šumy nebo například zastíněním scény po kratší dobu, musíme použít vhodného algoritmu pro jejich nalezení. Takto extrémní hodnoty by nám totiž i v malém počtu mohli vypočtenou amplitudu ovlivnit.

Pro nalezení výše zmíněných extrémů bylo použito algoritmu Local Outlier Factor (dále pouze LOF). Tento algoritmus složí k nalezení tzv. outlierů, neboli hodnot výrazně vybočujících z okolních hodnot. Abych nemusel celý tento poměrně složitý algoritmus programovat od začátku, byla jako základ použita implementace LOF od uživatelem *bkaluza* z portálu *github.com* [11]. Kód však bylo nutné přepsat z jazyku Java do jazyku C#, ve kterém je psána má aplikace. Spolu s tím byly i provedeny další menší změny, aby se dal algoritmus aplikovat na mnou požadované data.

LOF počítá tzv. skóre jednotlivých bodů, které reprezentuje vzdálenost od uvažované okolí (v našem případě bylo zvoleno okolí o velikosti 20). Čím je toto skóre vyšší tím výraznější outlier daný bod je. Na uživateli potom je, aby rozhodl, které hodnoty bude za outliery považovat. V našem případě považujeme za outliery všechny hodnoty se skórem $>= 2$. Všechny hodnoty označeny jako outlier nebudou uvažovány do výpočtu amplitudy. Na obrázku REFFF můžeme vidět graf s reálnými hodnotami polohy a jejich skóre určené LOF algoritmem.



Obrázek 6.10. Ukázka vypočteného skóre LOF algoritmem.

Úplně nakonec je spočtena amplituda, jakožto aritmetický průměr z 5% nejvyšších hodnot polohy v absolutní hodnotě. Pokud je těmito hodnotami vybrán outlier označený z předchozího kroku není uvažován a místo toho se vezme další hodnota v pořadí, tak aby vždy skutečně bylo vybráno 5% hodnot.

Celý tento výpočet, jakožto i předchozí kroky stojí na dostatečném počtu dat. Klíčová je především délka sekvence snímků. Čím delší bude, tím větší je šance správného určení amplitudy i při netrefení se do stroboskopické vzorkovací frekvence. Právě proto je raději obětován počet pixelů za množství snímků v sekvenci.

Kapitola 7
PC aplikace

Protože jde v této práci především o ověření možností detekce vibrací pomocí kamerového systému, probíhá veškeré zpracování dat v řídícím počítači. Tímto způsobem je jednodušší vyzkoušet různé algoritmy pro zpracování dat a zároveň máme lepší přístup k průběhu zpracování i výsledným datům. Za tímto účelem bylo třeba vypracovat aplikaci běžící na řídícím počítači. Tato aplikace se musí skládat ze dvou základních částí. První sloužící ke komunikaci s vývojovým kitem a druhá k samotnému zpracování dat.

Důležité rozhodnutí bylo v jakém programovacím jazyce aplikaci implementovat. V úvahu přidali hned 3 možnosti. Na prvním místě se pro mne nabízela Java. Z objektově orientovaných jazyků s ní mám nejvíce zkušeností. Ovšem neexistující standardní knihovna pro obsluhu COM portu mne od jejího použití odradila. Jako další dvě možnosti se potom nabízely jazyky C++ a C#. Z těchto dvou nakonec padla volba na jazyk C# s frameworkem .NET 4.6.2. Tato kombinace sice neumožně takovou volnost jako C++, ale zase je pro programátora o něco přívětivější. Velkou výhodou je hlavně garbage collector. Výhodný je i široký rozsah standardních knihoven, obsahující nástroje pro obsluhu COM portu i práci s bitmapovými obrázky. Protože jde především o testovací aplikaci není nutné použité algoritmy nějak extrémně optimalizovat, jak po paměťové tak časové stránce. Jazyk C# je také velmi podobný Javě. Možná i díky tomu se pro mne nejevilo jako zásadní problém ani to, že jsem v něm implementoval poprvé.

Vzhledem k omezeným časovým možnost a hlavnímu účelu aplikace byly udělány i jisté implementační kompromisy. Především nejsou ošetřeny všechny nestandardní stavy aplikace. S menší úmyslnou snahou uživatele ji tedy není velký problém shodit. Jelikož je ale určena především na testování naměřených dat a ne na běžné používání (například studenty), není to takový problém. Dokonalé ošetření všech nestandardních stavů by zabralo nemalé množství času, které bylo raději investováno do rozšiřování částí aplikaci skutečně rozvíjející obsah práce.

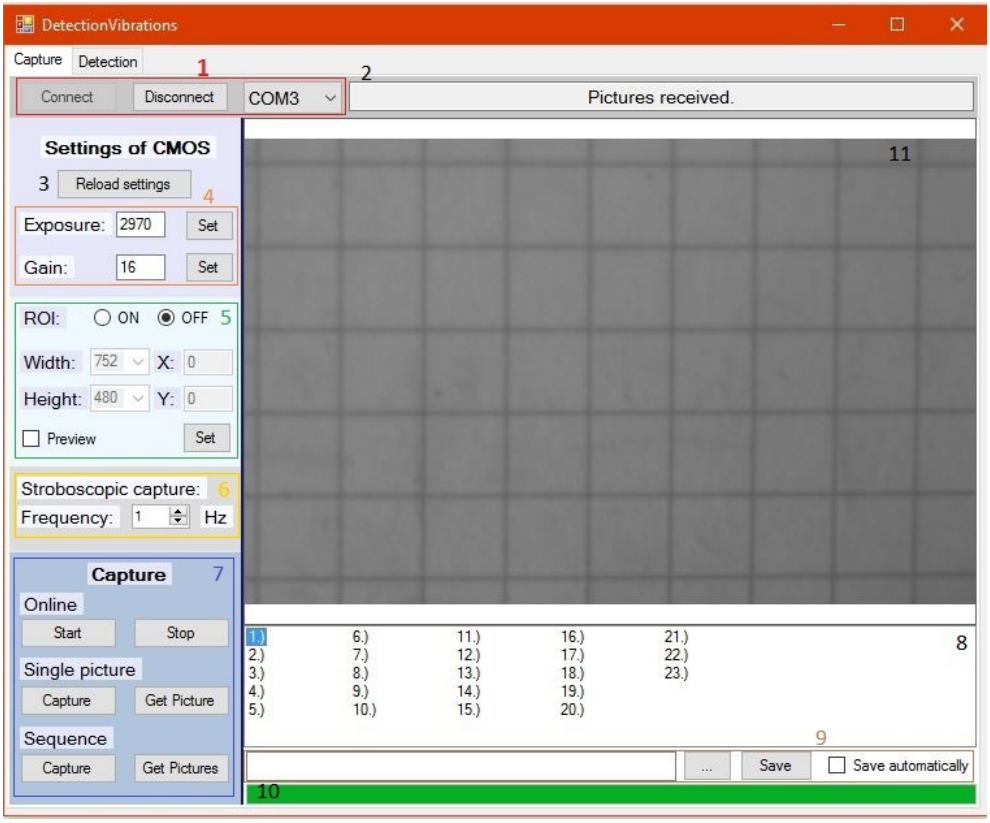
7.1 Snímací část aplikace

První část aplikace zajišťuje komunikaci s vývojovým kitem. Umožňuje posílání příkazů pro FW a zároveň přijímání dat. Další důležitou věcí je potom možnost ukládání přijatých snímku pro další zpracování. Možnosti této části aplikace jsou podrobněji popsány při popisu GUI dle obrázku 7.1.

1. **Spojení s vývojovým kitem**

   V této části se nacházejí 3 prvky. Zleva jsou to tlačítko *Connect* sloužící k vytvoření spojení aplikace a vývojového kitu. Následuje tlačítko *Disconnect* sloužící k ukončení spojení. Poslední je select box sloužící k výběru portu, kterým je kit připojen.
2. **Status bar**

   Status bar slouží k informování uživatele o probíhajících nebo právě provedených operacích.



Obrázek 7.1. GUI části aplikace určené pro snímání.

3. **Načtení registrů CMOS**

   Tlačítko *Reload settings* slouží k načtení konfiguračních registrů CMOS senzoru a zobrazení aktuálních informací v GUI.

4. **Nastavování světlosti obrazu**

   V rámci aplikace jsou dvě možnosti jak ovlivnit světlost obrazu. První je změnou doby expozice (*Exposure*), druhou potom změna analogového gainu (*Gain*). Tyto dvě možnosti jsou v řádcích pod sebou. Zleva je potom v řádku label označující o jaký parametr jde, text box na zapsání požadované hodnoty a tlačítko *Set* provádějící odeslání požadavku o nastavení do FW.

5. **Nastavování ROI**

   Ovládací prvky v této oblasti slouží k nastavování ROI. První dva radio buttony označené jako *ON*/*OFF* slouží k povolení a zakázání ROI módu. Pokud je zvoleno *OFF* není možné měnit velikost snímku. Ta je automaticky nastavena na plnou velikost. Pokud je zvoleno *ON* zpřístupní se další možnosti a lze měnit velikost snímku. V oblasti pod radio buttony se nachází 2 select boxy označené *Width* a *Height* a 2 text boxy označené *X* a *Y*. Select boxy umožní výběr šířky (*Width*) a výšky (*Height*) snímku. Text boxy potom slouží k nastavení posunu snímku v horizontálním (*X*) a vertikálním (*Y*) směru v počtu pixelů. V dolní levé části se nachází check box označený jako *Preview*. Po jeho zaškrtnutí se zobrazí červený rámeček na momentálně zobrazeném obrázku (11) o velikosti odpovídající aktuálním hodnotám v ostatních prvcích. Tlačítko Set potom samozřejmě slouží k nastavení vybraných hodnot do senzoru.

6. **Výběr stroboskopické frekvence**

   Hodnota nastavená v této oblasti se použije pouze při odběru sekvence snímků. Je to hodnota, pro kterou je vypočtena odpovídající stroboskopická frekvence, na které je následně snímkováno.

7. **Odběr snímků**

   Celá tato oblast GUI slouží k zahájení odběru snímků a jejich následného odesílání. Dala by se ještě rozdělit na 3 menší částí. První je označena jako *Online*. Ta obsahuje tlačítka *Start* a *Stop*. Tlačítko *Start* zahájí kontinuální odběr snímků, které jsou zobrazovány v GUI (11). Při spuštěném kontinuálním odběru lze měnit dobu expozice a analogový gain, nikoli však velikost snímku. Slouží tedy především k vyladění světelnosti a zaostření objektivu. Tlačítko *Stop* potom logicky zastaví kontinuální odběr snímků.

   Další podoblasti jsou označeny *Single picture* a *Sequence*. Tlačítka jsou téměř totožná. V *Single picture* to jsou *Capture* a *Get picture*. *Capture* slouží k zahájení odběru jednoho snímku. *Get picture* potom k zahájení přenosu jednoho snímku do PC. V *Sequence* podoblasti je to velmi podobné. *Capture* tlačítko slouží k zahájí odběru sekvence snímků na stroboskopické frekvenci odpovídající frekvenci nastavené v (6). Tlačítko *Get pictures* potom slouží k zahájení odesílání celé sekvence snímků.

8. **Výběr snímků**

   Po přijmutí sekvence snímků se v této oblasti objeví číselný seznam přijatých snímků. Čísla jsou přiřazována v tom pořadí, v jakém byly snímky přijaty. Výběrem čísla v této oblasti zobrazíme odpovídající snímek v oblasti (11).

9. **Ukládání snímku**

   Možnosti pro ukládání právě zobrazených snímků jsou v této oblasti. Pokud jednotlivé prvky projdeme zleva, jsou to následující. První je text box zobrazující aktuálně vybranou složku pro ukládání snímků. Druhé je tlačítko ... vyvolávající po zmáčknutí dialogové okno pro výběr lokace uložení. Třetí je tlačítko *Save*, po jehož zmáčknutí je provedeno samotné uložení. Uloženy jsou vždy aktuálně načtené snímky. Pokud tedy byla naposledy načtena sekvence, bude uložena sekvence snímků. A pokud byl naposledy načten jednotlivý snímek, bude uložen jen ten. Posledním prvkem je check box označený jako *Save automatically*. Po jeho zaškrtnutí je ukládání právě přijímaných snímků prováděno automaticky do vybrané složky.

   Struktura ukládání je následující. Nejprve je, pokud ještě neexistuje, vytvořena složka ve formátu DD_MM_YYYY, kde DD je aktuální den, MM aktuální měsíc a YYYY aktuální rok. Pokud je ukládán jednotlivý snímek, je uložen do této složky ve formátu hh_mm_ss_uuuuuu_EEEEus.bmp, kde hh je hodina, mm minuty, ss sekundy a uuuuuu μs odpovídající času uložení snímku. Poslední část EEEE je nastavená doba expozice následovaná jednotkou času, ve které je expozice uváděna. Pokud je ukládána sekvence snímků, je postup delší o jeden krok. Nejprve je vytvořena složka ve formátu SEQ_hh_mm_ss_FHz_EEEEus, kde všechny části mají stejný význam jako u ukládání jednoho snímku. Nová je část FHz, ve které F značí frekvenci, na jejíž odpovídající stroboskopické frekvenci byl odběr prováděn. Jednotlivé snímky jsou do takto vytvořené složky ukládány ve formátu N_uuuuuu.bmp, kde N je číslo snímku a uuuuuu čas uložení snímku v μs. Pro zachování správné funkčnosti druhé části aplikace je důležité zachovávat tento formát.

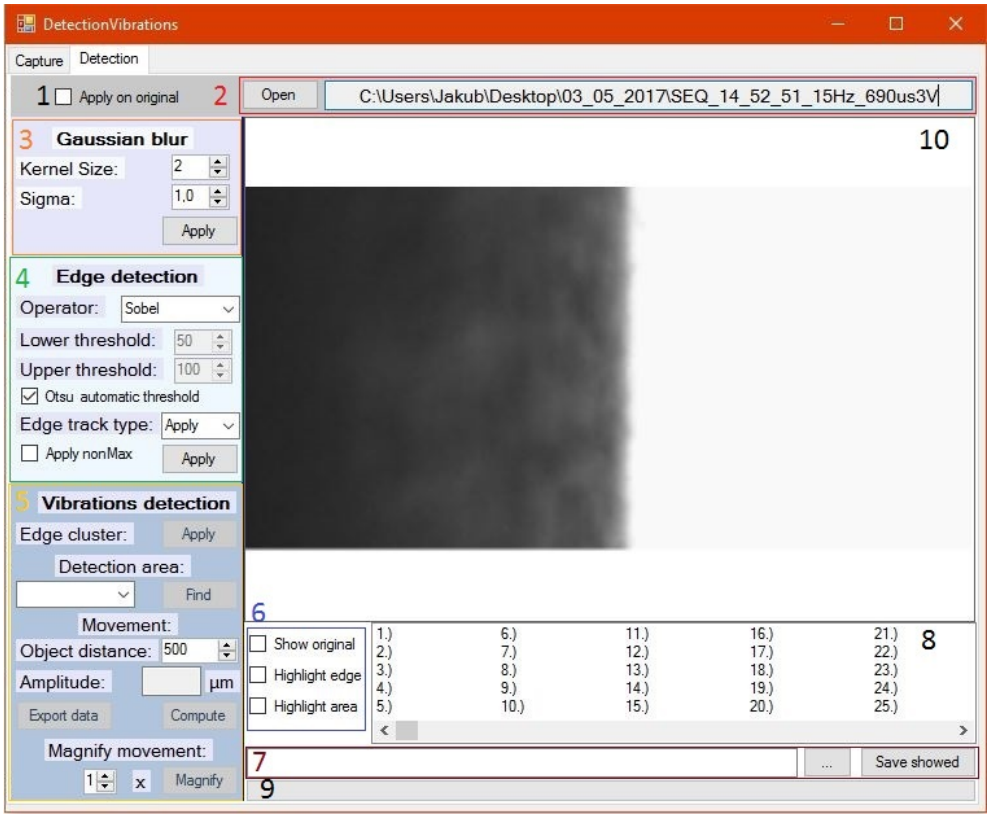
10. **Progress bar**

    Progress bar slouží k informování uživatele o přibližném časovém průběhu déle trvajících operací. Mezi ně patří například odběr sekvence snímku nebo přijímání sekvence snímků.

11. **Snímek**
    Zde je zobrazen aktuálně vybraný snímek. Při přijmutí sekvence snímků je možné zobrazený snímek měnit výběrem v oblasti (8).

7.2 Detekční část aplikace

Druhá část aplikace slouží ke zpracování nasnímaných dat. Obsahuje rozhraní pro postupnou aplikaci implementovaných algoritmů a umožňuje ukládání zpracovaných snímků a export výsledných dat ve formátu .xlsx (formát Microsoft Excel). Podrobný popis je znovu proveden při popisu GUI této části aplikace dle obrázku 7.2.



Obrázek 7.2. GUI části aplikace určené pro zpracování dat.

1. **Aplikování na originální snímky**
   Check box *Apply on original* umožňuje určit, na kterou verzi snímků se budou filtry aplikovat. Po zaškrtnutí se budou aplikovat na originálně otevřené snímky. Pokud zaškrtnut není (výchozí stav) jsou filtry aplikovány na současnou verzi snímků (například již s aplikovanými filtry). Tato možnost je tu především pro urychlení práce v případě aplikace špatně nastaveného filtru. Není potom nutné znovu otvírat složku se snímky.
2. **Otevření obrázků**
   V této oblasti se nachází pro uživatele důležité tlačítko *Open*. To vyvolá dialogové okno pro výběr adresáře. Po výběru adresáře jsou z něj načteny veškeré bitmapové soubory do aplikace (zobrazí se v části 8). Text box potom slouží k výpisu cesty k aktuálně otevřené složce.

3. **Gaussův filtr**

   Tato oblast aplikace slouží k nastavení a aplikaci Gaussova filtru 6.2.1. Ten má dva parametry. Prvním je velikost konvoluční masky *Kernel Size* ve tvaru $(2k+1)$x$(2k+1)$, kde volíme velikost hodnoty k. Druhým parametrem je potom hodnota σ *Sigma*. Po zvolení těchto dvou hodnot provedeme aplikaci filtru na všechny otevřené snímky pomocí tlačítka *Apply*.

4. **Hranová detekce**

   Oblast aplikace zabývající se hranovou detekcí je poměrně rozsáhlá. Možnosti v ní můžeme rozdělit do šesti řádků. První obsahuje výběr operátoru *Operator*. Další tři řádky se zabývají volbou prahů 6.2.4. Důležitý je stav check boxu *Otsu automatic threshold*. Pokud je zaškrtnutý, jsou prahy voleny automaticky pomocí Otsuho metody prahování 6.2.5. Pokud ovšem není zaškrtnutý, umožní se ruční nastavení hodnot dolního prahu *Lower threshold* a horního prahu *Upper threshold*.

   Nastavení hranové detekce má ještě další dvě možnosti. Jednou je nastavení typu hysterezního sledování hran *Edge track type* 6.2.6. Může být povoleno (možnost Apply) nebo vypnuto ve dvou různých módech. Všechny slabé hrany mohou být zachovány (možnost Keep) nebo naopak potlačeny (možnost Remove). Poslední možností je aplikace Non-maximum potlačení 6.2.3 řízená check boxem *Apply nonMax*. Při zaškrtnutí je aplikováno, jinak aplikováno není.

5. **Detekce vibrací**

   Kroky potřebné pro spočtení amplitudy po aplikaci hranové detekce lze provést v této oblasti aplikace. Nejprve je potřeba provést clustering hran 6.3 pomocí tlačítka *Apply* nacházejícího se vedle textu *Edge cluster*. Po provedení clusteringu je možné vybrat sledovanou hran v select boxu pod textem *Detection area*. Po vybrání hrany je oblast pro sledování vibrací 6.3.2 určena pomocí tlačítka *Find*.

   Pro výpočet amplitudy je potřeba zadat vzdálenost objektu od čočky v μm do boxu u textu *Object distance*. Amplituda je následně spočtena 6.4 stisknutím tlačítka *Compute*. Vypočítaná amplituda je následně zobrazen v text boxu označeného textem *Amplitude*.

   Všechny spočtené hodnoty průběhu je následně možné vyexportovat ve formátu .xlsx tlačítkem *Export data*. K exportu dat byla použita externí knihovna ClosedXML určená pro práci s Excelem [12].

   Poslední je potom část *Magnify movement*. Ta slouží ke zvětšení pohybu v právě otevřených snímcích. Tato možnost byla vytvořena nad rámec práce pouze pro zajímavost. Při aplikaci ve skutečnosti pouze zvyšuje jasové rozdíly mezi pixely v po sobě jdoucích snímcích ve vybrané oblasti. Hodí se tedy pouze pro velmi malé pohyby, které již téměř nejsme schopni zachytit lidským okem. Při větším faktoru zvětšení nebo při zvětšování většího pohybu se začne obraz rozpadat.

6. **Možnosti zobrazení**

   V aplikaci je možné přepínat mezi zobrazením různých verzí aktuálně načtených obrázků pomocí tří check boxů. První z nich *Show original* přepíná mezi zobrazením originálního snímku a aktuálního upraveného snímku. *Highlight edge* vykreslí právě vybranou hranu červenou barvou přes zobrazený snímek. *Highlight area* obdobně vykreslí oblast detekce pro vybranou hranu zelenou barvou přes zobrazený snímek.

7. **Ukládání snímků**

   Část ukládání snímků funguje totožně, jako ve snímací části aplikace. Jenom ukládá vždy pouze právě zobrazený stav snímku a v názvu není doba expozice. Nelze tedy uložit všechny načtené snímky najednou. Slouží tedy především k ukládání ukázkových snímků.

8. **Výběr snímků**

   Po otevření složky v této oblasti objeví číselný seznam snímků. Čísla jsou přiřazována v tom pořadí, v jakém byly snímky načteny. Výběrem čísla v této oblasti zobrazíme odpovídající snímek v oblasti (10).
9. **Progress bar**

   Progress bar slouží k informování uživatele o přibližném časovém průběhu déle trvajících operací. Mezi ně patří například aplikace Gaussova filtru nebo hranové detektoru.
10. **Snímek**

    Zde je zobrazen aktuálně vybraný snímek. Při přijmutí sekvence snímků je možné zobrazený snímek měnit výběrem v oblasti (8).

Kapitola 8
Ověření funkčnosti měření amplitudy

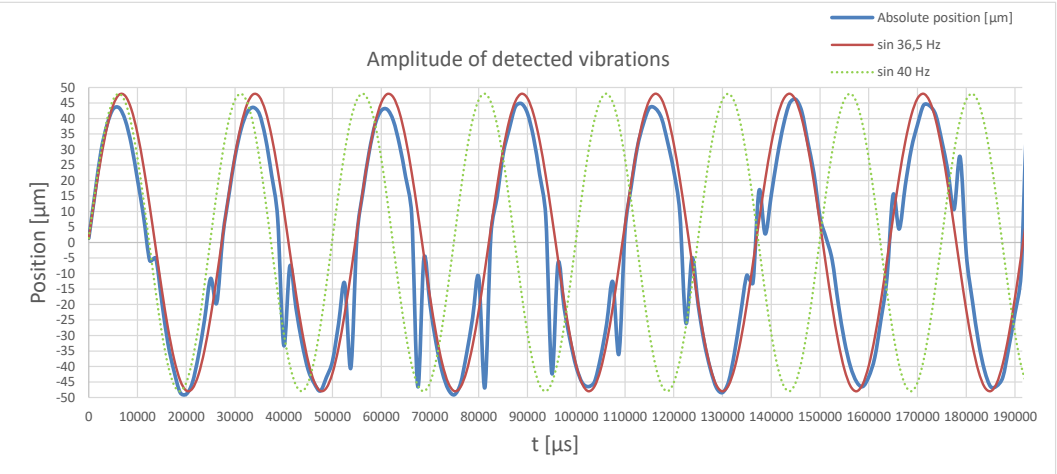
Hlavním výstupem detekce vibrací je jejich amplituda. Abychom ověřili funkčnost použitých postupů, je nutné provést reálné měření. Měřením ověříme reálnou funkčnost stroboskopického snímkování a následně výpočet amplitudy hledaných vibrací. Nasnímaná data jsou na přiloženém CD.

8.1 Měření na shakeru

Pro ověření funkčnosti našeho detektoru vibrací by bylo příhodné měřit na přístroji, u kterého lze ovládat amplitudu i frekvenci vibrací. To lze do jisté míry i změnou otáček elektromotoru, ale nelze to dělat velmi přesně. Proto bylo rozhodnuto použít permanent magnet shaker LDS V406. Pomocí generátoru funkcí a budiče lze přesně řídit průběh pohybu shakeru. Jelikož chceme simulovat vibrace, byl volen sinusový průběh o různých frekvencích a amplitudě. Bohužel je amplituda pohybu závislá nejen na amplitudě signálu ale i na jeho frekvenci. Tato závislost navíc není lineární, takže nejsme schopni výpočtem určit skutečnou amplitudu.

Frekvenčně jsme se při měření pohybovali mezi 10-40 Hz. Amplituda sinusového průběhu z generátoru se potom pohybovala mezi 5-100 mV. Menší amplituda než-li 5 mV bohužel nešla na generátoru nastavit a musel jsem se spokojit s těmito hodnotami.

Funkčnost stroboskopického snímkování můžeme nejlépe ukázat na nejvyšší snímané frekvenci 40 Hz. Ukázka je omezena pouze na část dat, protože celý průběh má dlouhou časovou osu. Perioda sinu je potom vykreslena na malý prostor a není dostatečně dobře vidět. Pro ukázku byly vybrány data snímaná při amplitudě signálu generátoru 50 mV 8.1.

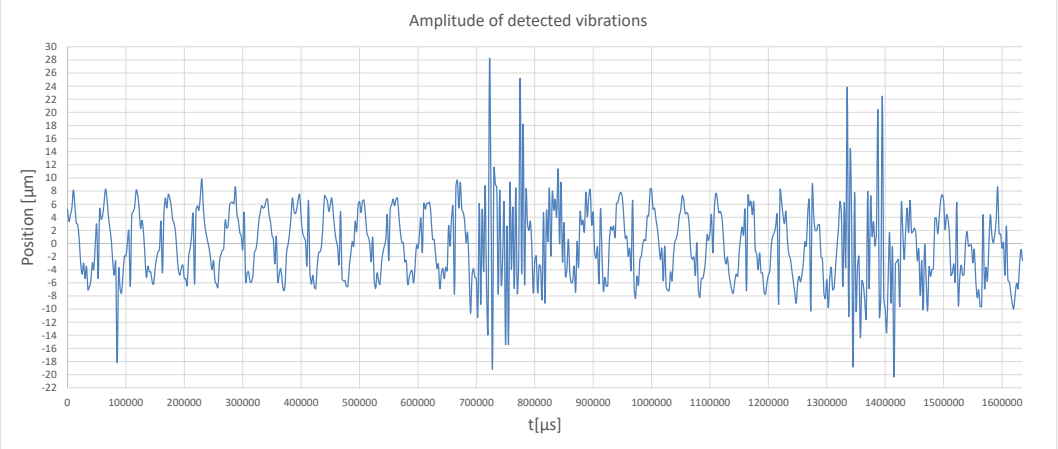


Obrázek 8.1. Průběh polohy při měření na shakeru při nastavení generátoru - frekvence 40 Hz, amplituda 50 mV. Pro ukázku jsou vykresleny i průběhy napočítaných sinusových průběhů.

Graf 8.1 je vykreslený v ekvivalentním stroboskopickém čase. Pokoušíme se tedy rekonstruovat signál bez splnění vzorkovacího teorému. Změřený průběh by měl být ekvivalentní průběhu získaného na snímkovací frekvenci 800 Hz. V grafu si můžeme všimnout hned několika věcí. První je asi to, že spočtený průběh polohy poměrně věrně kopíruje sinusový průběh. Vidíme ale také, že má blíže k sinu o frekvenci 36.5 Hz. Se sinusovým průběhem o frekvenci 40 Hz se po krátké době začne rozcházet. Důvody můžou být hned tři. Prvním je nepřesnost shakeru. Pohybujeme se totiž stále u jeho téměř minimálních frekvenčních hodnot. Je tedy možné, že se frekvence o něco málo rozchází. Druhou možností je nepřesnost stroboskopického snímkování. Dle signálů senzoru měřených na osciloskopu 5.3.3 vše vypadalo dobře, ale je možné, že při vyšších frekvencích je přesnost nedostatečná. Poslední možností je prostý šum měření.

I přes menší nepřesnosti je sinusový průběh dobře viditelný. Není tedy nejmenší problém spočítat amplitudu vibrací, která byla v tomto případě změřena na 48 μm.

Další důležitým faktorem je určitě citlivost. Pro ukázku můžeme tentokrát zvolit jinou frekvenci. Následující průběh byl změřen při nastavení generátoru na frekvenci 20 Hz a amplitudu 5 mV 8.2. Tentokrát také necháme vykreslený celý změřený průběh.

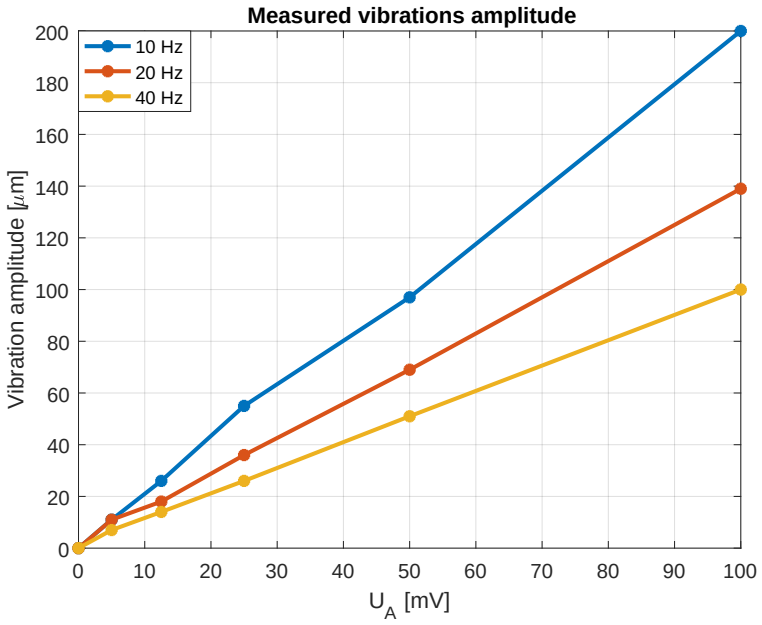


Obrázek 8.2. Průběh polohy při měření na shakeru při nastavení generátoru - frekvence 20 Hz, amplituda 5 mV.

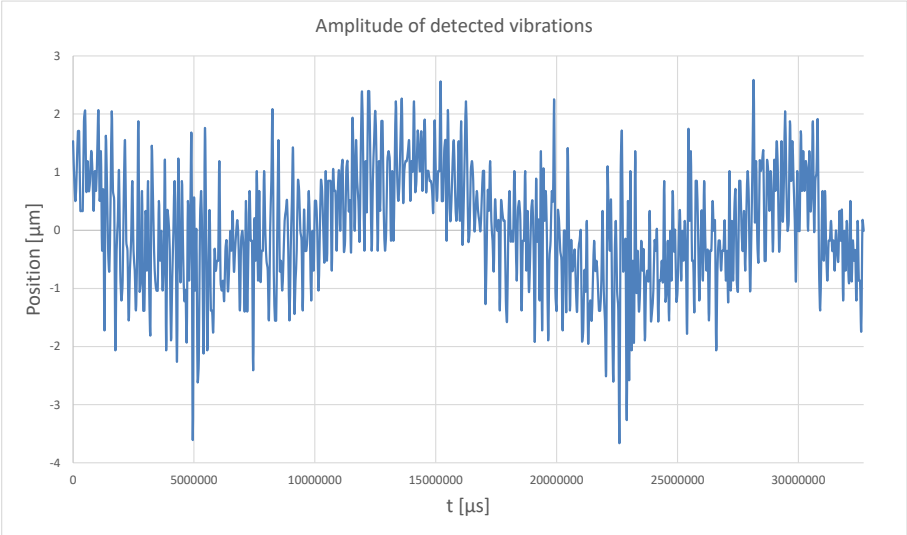
Z grafu 8.2 tentokrát vidíme, že je průběh zdánlivě mnohem více zatížen šumem. Vypadá to tak, především protože se pohybujeme v přibližně 5x menším měřítku než v předchozím případě. Filtrace pomocí LOF algoritmu ovšem splnila svoji roli a amplituda vibrací byla spočtena jako 10 μm.

Ostatní změřená data jsou na přiloženém CD. Pro zajímavost si ještě můžeme vykreslit změřenou závislost amplitudy vibrací na frekvenci a amplitudě signálu generátoru 8.3.

Závislost 8.3 měřené amplitudy vibrací na amplitudě signálu generátoru je téměř lineární. To odpovídá očekávání. Vidíme ale, že skutečně existuje poměrně silná frekvenční závislost měřené amplitudy na frekvenci signálu generátoru. To je pravděpodobně způsobeno tím, že se pohybujeme velmi nízko ve frekvenčním rozsahu shakeru a nejsme tedy v jeho typické pracovní oblasti.



Obrázek 8.3. Závislost amplitudy vibrací na frekvenci a amplitudě signálu generátoru.



Obrázek 8.4. Zachycení aliasingového sinusového průběhu při vypnutém výstupu generátoru.

Poslední velmi zajímavá data zachycená na shakeru pochází ze snímání při vypnutém výstupu generátoru. Snímáno bylo bez stroboskopického snímkování na 20 Hz (tedy 20 snímků za sekundu). Průběh absolutní polohy je vykreslen na grafu 8.4.

V průběhu změřeném při vypnutém výstupu generátoru 8.4 můžeme vidět zajímavý úkaz. Je to zachycený aliasingový sinusový průběh. To znamená, že i přes vypnutý výstup generátoru se nějaký signál z budiče do shakeru dostal. Byli jsme tedy schopni změřit i šumový signál přicházející do shakeru. Pravděpodobně se potom jedná o šumový signál způsobený síťovým napětím. Aplituda toho signálu byla spočtena jako 2 μm.

Měřením na shakeru jsme dostatečně prověřili funkčnost měření amplitudy vibrací při známé frekvenci vibrací. Prověřili jsme i citlivost měření, kdy jsme byli schopni změřit i šum přicházející do shakeru. S citlivostí jsme se tedy přiblížili k jednotkám μm. Citlivost by mohla být ještě vyšší v případě snímání hrany z bližší vzdálenosti. V našem případě

byla snímána ze vzdálenosti 400 mm. Minimální vzdálenost měřeného objektu od čočky byla spočtena na 250 mm. Lze tedy předpokládat, že citlivost měření by se při přiblížení objektu ještě zlepšila.

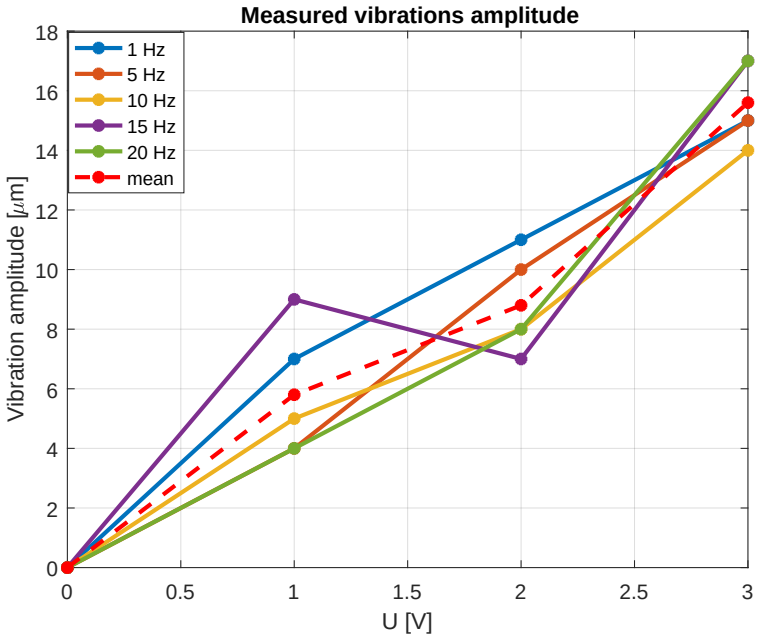
Celkově můžeme považovat měření na shakeru za velmi přínosné. Dosažené výsledky jsou pro nás potřebnou referencí pro určení funkčnosti měřící sestavy. Měření dopadlo nad mé očekávání a myslím, že je možné ho považovat za úspěšné.

8.2 Měření na elektromotoru

Druhé měření bylo provedeno na malém elektromotoru. Šlo zde o ověření schopnosti výpočtu amplitudy vibrací při snímání na frekvenci neodpovídající frekvenci vibrací. Snímáno bylo při různých napětí na motoru a na různých snímkovacích frekvencích.

Protože bylo snímáno na frekvencích neodpovídajících frekvenci vibrací motoru, postrádá smysl uvádět zde naměřené průběhy polohy. Maximálně je v těchto průbězích vidět aliasingový sinusový průběh. V případě zájmu jsou všechna naměřená data na přiloženém CD.

Na následujícím grafu 8.5 je zobrazena závislost změřené amplitudy vibrací na napájecím napětí DC motoru. Je vykresleno několik průběhů, kde každý z nich odpovídá snímání na stroboskopické frekvenci odpovídající uvedené frekvenci. Navíc je uveden průběh tvořený průměrnými hodnotami na dané frekvenci.



Obrázek 8.5. Závislost amplitudy vibrací na napájecím napětí DC motoru. Měřeno na různých frekvencích snímkování.

Z grafu je viditelné, že především při snímání na stroboskopické frekvenci odpovídající 15 Hz nedopadl výpočet amplitudy vibrací dle očekávání. Amplituda vibrací při napájení DC motoru 1 V byla spočtena vyšší než-li při napájení 2 V. Chybný výpočet byl způsobem příliš velkým šumovým zatížením v případě napájení 1 V. Šumové zatížení totiž bylo natolik velké, že si sním již filtrovací algoritmy nedokázaly poradit. Velké změny jasových hodnot způsobené světelným šumem, totiž byly shluknuty ve velkém množství blízko

sebe. Používaný LOF algoritmus je ovšem nastaven na odchytávání outlierů výrazně vystupujících z blízkého okolí. Při větším shluku takovýchto hodnot již nevystupují ze svého blízkého okolí a nejsou tedy označeny za outliery.

Snímání na ostatních frekvencích naopak dopadlo dle očekávání. Důležitá je viditelná přímá úměrnost amplitudy vibrací na velikost napájecího napětí DC motoru. Otáčky DC motoru totiž mají lineární závislost na napájecím napětí.

Přesnost výpočtu amplitudy při snímání na frekvenci neodpovídající frekvenci vibrací není tak přesná jako při snímání na stroboskopické frekvenci. Tuto nepřesnost je možné do určité míry eliminovat provedením více odběrů dat na různých snímkovacích frekvencích. Jejich následným zprůměrováním dostaneme poměrně dobrou reprezentaci velikosti amplitudy vibrací.

Kapitola 9
Závěr

V rámci této bakalářské práce se podařilo splnit všechny body zadání. Hlavním cílem bylo ověření možností detekce vibrací pomocí informace z obrazového CMOS senzor. K tomu bylo potřeba vytvořit systém pro snímání vibrací, skládající se především z CMOS senzoru řízeného mikrořadičem z řady STM32. Následně bylo nutné implementovat FW potřebný k řízení vytvořeného kamerového systému. V neposlední řadě byla vytvořena aplikace pro nadřazené PC určená pro zpracování obrazových dat.

Nejprve bylo nutné navrhnout měřící sestavu. Obrazový CMOS senzor byl vybrán typu global shutteru, kvůli snímání rychle se pohybujících objektů. Senzor je řízen vývojovým kitem STM32F429, který byl zvolen především kvůli velikosti SDRAM. Snímaný objekt je kamerovým systémem snímán ve směru kolmém ke směru vibrací. Za snímaný objekt musí být umístěn backlight pro zajištění dostatku světla při krátké expozici.

Klíčovým obsahem FW vývojového kitu je především schopnost řízení odběru snímků v přesně definované okamžiky. To umožňuje stroboskopické snímkování, díky kterému je možné detekovat vibrace na frekvenci vyšší nežli při běžném snímkování. Tímto způsobem jsme schopni simulovat snímkovací frekvence více než 10x vyšší, než je maximální snímkovací frekvence senzoru (samozřejmě jen pro periodické pohyby). FW tedy umožňuje odběr sekvence snímků na přesně definované frekvenci. Dále je tu i možnost jednotlivého odběru snímku. Snímky jsou do paměti SDRAM přenášeny přes sběrnici DCMI pomocí DMA. Nastavování CMOS senzoru probíhá přes I^2C sběrnici. Za tímto účelem byla použita knihovna pro CMOS senzor vytvořená v rámci BP *Zpracování obrazu mikrořadiči pro mnohokanálové měření polohy* [6].

Komunikace mezi vývojovým kitem a řídícím PC je řešena pomocí USB CDC knihovny vytvářející Vitual COM port. Kvůli zajištění spolehlivosti komunikace bylo vytvořeno řízení toku dat na aplikační úrovni v podobě řízení toku paketů. Tímto způsobem se podařilo maximální rychlost Virtual COM portu přes full-speed USB dostat k hodnotám 800 kB/s. Nadřazené PC posílá požadavky do FW prostřednictvím vytvořené aplikace. Z nadřazeného PC je potom možné řídit odběru snímků, posílání dat i nastavování senzoru.

Samotná detekce vibrací je založena na hledání jasových změn mezi snímky ve vybrané oblasti snímku. Tato oblast je určena pomocí hranového detektoru a následných metod clusteringu. Pro detekci vibrací je tedy vždy sledována hrana s jasným přechod objektu do pozadí. Pro detekci hran byl zvolen Cannyho hranový detektor s možností volby jednotlivých kroků a volby operátoru. V získané oblasti detekce jsou následně spočteny jasové změny mezi snímky přes řádky hrany. Z nich je vypočítána jasová změna reprezentující pohyb hrany. Tu je možné přepočítat na změnu polohy v pixelech, kterou je možné pomocí zobrazovací rovnice přepočítat na změnu polohy v jednotkách vzdálenosti. Z průběhu polohy po filtraci LOF algoritmem je následně spočtena amplituda vibrací.

Vytvořená aplikace pro nadřazené PC byla implementována v jazyku C# framework .NET 4.6.2. Umožňuje komunikaci s vývojovým kitem a především zpracování obrazových dat. Samozřejmostí je ukládání snímků z aplikace do vybrané složky. Spočtená data je pak možné exportovat ve formátu .xlsx (Microsoft Excel).

Při měření na přesně řízeném shakeru jsem byl schopen ověřit funkčnost stroboskopického snímkování. Podařilo se simulovat snímkovací frekvence více než 10x vyšší než je maximální snímkovací frekvence senzoru. Při použití stroboskopického snímkování se přesnost měření amplitudy pohybovala v nižších jednotkách μm. Při měření na DC motoru bylo ověřeno, že lze měřit amplitudu vibrací i mimo stroboskopické frekvence. Přesnost sice není tak vysoká, jako v případě stroboskopického snímkování, ale to lze zlepšit opakovaným odběrem sekvence snímků. Citlivost detekce je momentálně taková, že není problém zachytit vibrace s amplitudou menší než 5 μm.

V rámci práce se tedy podařilo ověřit, že detekce vibrací pomocí informace z obrazového senzoru je možná. Podařilo se naměřit očekávána dat, jak v případě měření na přesně řízeném shakeru, tak při měření na DC motoru. Definované měřící prostředí je ovšem velmi idealizované a v praxi s ním není možné úplně počítat. Dosažené výsledky jsou ale dostatečně dobré, aby mělo smysl systém dále rozvíjet a pokusit se ho dostat do prakticky použitelného stavu.

Prvním důležitým krokem k lepší použitelnosti by bylo přenesení veškerého zpracování dat do mikrořadiče. Celý měřící systém by pak mohl fungovat jako embedded systém. Dále je potřeba řešit vysokou citlivost na světelný šum. Bylo by vhodné přidat do systém způsob normalizace jasu snímků. V nejednodušším případě by to mohla být fotodioda. Díky informaci z ní by bylo možné řídit přímo analogový gain senzoru a tím si ušetřit normalizaci jasu snímku po jeho uložení. Velkým problémem by také bylo samotné uchycení systému. Při snímání vibrací velkých průmyslových strojů by bylo potřebné zajistit izolování měřícího systému od všudypřítomných vibrací.

K praktické aplikaci podobného systému je nutné vyřešit ještě velké množství dílčích problémů. Věřím však, že poznatky z této práce jsou dostačující jako důvod k pokračování vývoje podobného systému.

Literatura

[1] Ripka, Pavel, Stanislav Ďaďo, Marcel Kreidl a Jiří Novák. *Senzory a převodníky*. Praha: Skripta ČVUT FEL, 2005. ISBN 80-01-03123-3.

[2] Fischer, Jan. *Optoelektronické senzory a videometrie*. Praha: Skripta ČVUT FEL, 2002. ISBN 80-01-02525-1.

[3] RED.COM, INC. *red.com: RED 101 (online)*. http://www.red.com/learn/red-101/. (cit. 2017-05-16).

[4] Toms, Daniel. *Spracování obrazu pro sledování optické stopy*. Praha: ČVUT FEL, Katedra měření, 2014. Bakalářská práce.

[5] Řípa, Radek. *Synchronizované obrazové snímače*. Praha: ČVUT FEL, Katedra měření, 2012. Diplomová práce.

[6] Novák, Tomáš. *Zpracování obrazu mikrořadiči pro mnohokanálové měření polohy*. Praha: ČVUT FEL, Katedra kybernetiky, 2013. Bakalářská práce.

[7] Dokoupil, Vojtěch. *STM32F429 Based CMOS Camera for Teaching Purposes*. Praha: ČVUT FEL, Katedra měření, 2015. Diplomová práce.

[8] STMicroelectronics. *RM0090, STM32F4 Reference manual, Rev. 13*. http://www.st.com.

[9] Davies, E.R. *Computer and Machine Visions: Theory, Algorithms, Practicalities*. 4. vyd. Oxford: Academic Press, 2012. ISBN 978-0-12-386908-1.

[10] Wu, Hao-Yu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand a William T. Freeman. Eulerian Video Magnification for Revealing Subtle Changes in the World. *ACM Transactions on Graphics (Proc. SIGGRAPH 2012)*. 2012, ročník 31, č. 4. http://people.csail.mit.edu/mrub/evm/.

[11] Kaluza, Bostjan. *Java implementation of Local Outlier Factor algorithm*. https://github.com/bkaluza/jlof. (cit. 2017-05-20).

[12] ClosedXML. *Library for manipulating Excel 2007/2010 files*. https://github.com/closedxml/closedxml. (cit. 2017-05-20).

[13] STMicroelectronics. *DS9405, STM32F429 Data, Rev. 9*. http://www.st.com.

[14] STMicroelectronics. *UM1670, User manual - STM32F429ZI, Rev. 2*. http://www.st.com.

[15] ONSemiconductor. *MT9V034 Datasheet, Rev. G*. http://www.alldatasheet.com/datasheet-pdf/pdf/759782/ONSEMI/MT9V034.html.

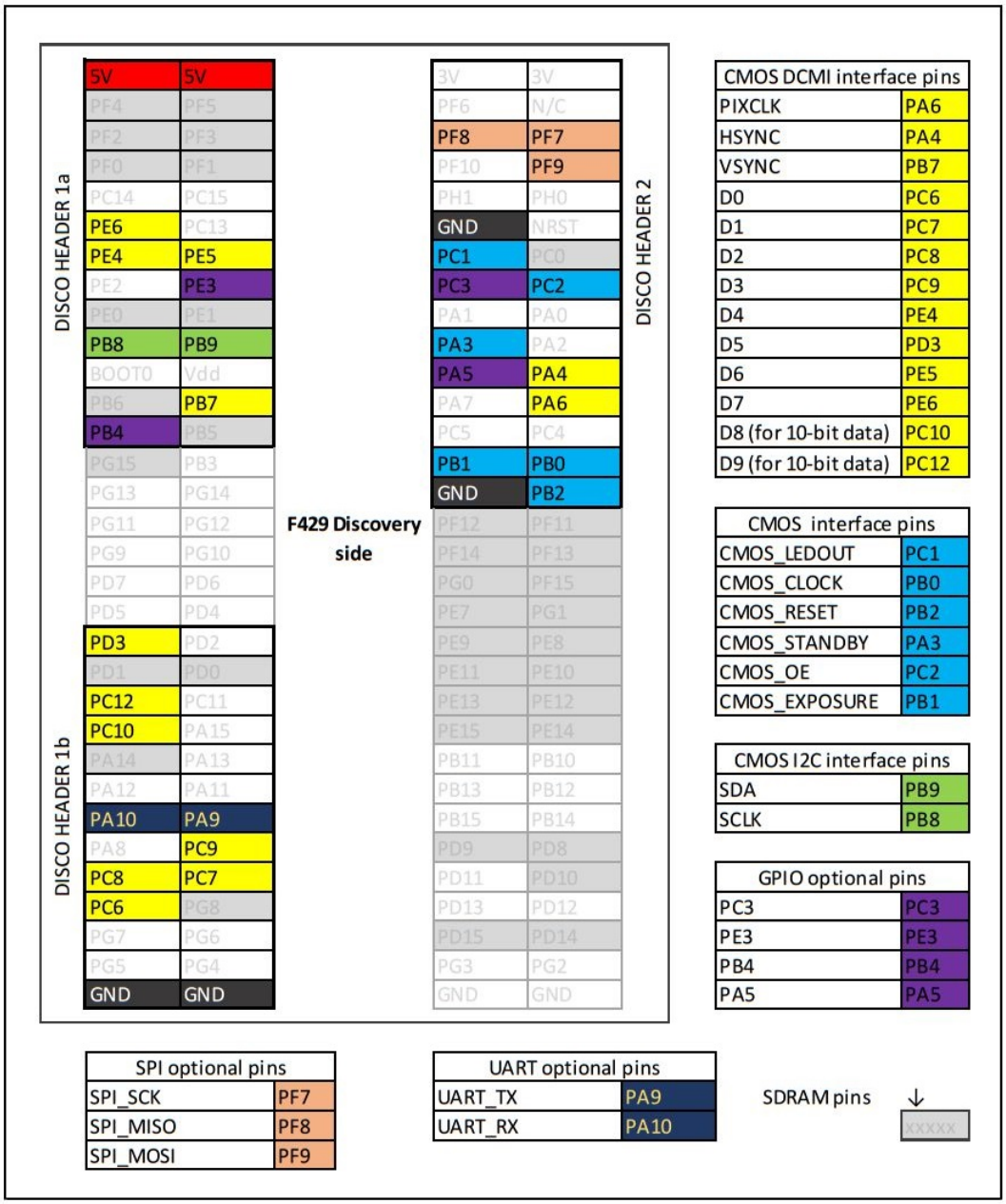
[16] Yiu, Joseph. *The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors*. 3. vyd. Oxford: Newnes, 2013. ISBN 978-0-12-408082-9.

Příloha A
Použité zkratky

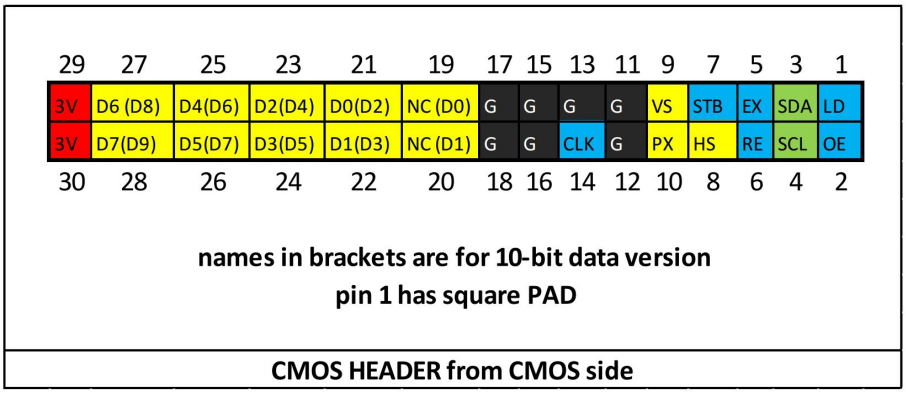
A/D převodník	Analogově digitální převodník.
AEC	Automatic Exposure Control (automatická kontrola doby expozice).
AGC	Automatic Gain Control (automatická kontrola zesílení).
ARM	Advanced RISC Machine. Typ architektury procesorů.
ASCII	American Standard Code for Information Interchange (americký standardní kód pro výměnu informací). Kódová tabulka definující znaky anglické abecedy a jiné používané znaky.
CCD	Charge-coupled device (zařízení s vázanými náboji). Typ obrazového senzoru.
CMOS	Complementary Metal–Oxide–Semiconductor (doplňující se kov-oxid-polovodič). Typ technologie používaný integrovanými obvody. V této práci se zkratka objevuje v podobě „CMOS senzor“, čímž je myšlen typ obrazového senzoru.
COM port	Communication port (komunikační port). Nejběžnější typ sériového rozhraní.
BP	Bakalářská práce.
DCMI	Digital Camera Interface. Paralelní rozhraní pro práci s digitálními obrazovými senzory.
DMA	Direct Memory Access (přímí přístup do paměti). Způsob přímého přenosu dat z periferie do operační paměti. Data neprochází přes procesor, čímž se šetří procesorový čas.
DP	Diplomová práce.
fpp	Zkratka frames per period (snímků za periodu).
fps	Zkratka frames per second (snímků za sekundu). Velmi často používaná jednotka snímkovací frekvence.
FW	FirmWare. Software pro řízení vestavěného systému.
HAL	Hardware abstraction layer (hardwarová abstraktní vrstva). Vrstva vytvářející jednotné rozhraní pro ovládání různě fungujícího HW.
I^2C	Inter-Integrated Circuit. Sériová sběrnice pro komunikace periferie s řídícím systémem.
LED	Light-Emitting Diode (dioda emitující světlo). Polovodičová součástka schopná vyzařovat světlo.
LOF	Local outlier factor. Algoritmus na hledání anomálních hodnot v sekvenci dat.
PC	Personal Computer (osobní počítač).
PCB	Printed Circuit Board (deska plošných spojů).
QVGA	Quarter Video Graphics Array. Pojem používaný pro rozlišení (320x240 px). Nabízí čtvrtinu obrazových bodů oproti maximu VGA.
RAM	Random Access Memory (paměť s náhodným přístupem). Typ paměti s přímím přístupem umožňující čtení i zápis.
ROI	Region of interest. Vybraná podmnožina vzorků z množiny dat.

SDRAM	Synchronous Dynamic Random Access Memory (synchronní dynamická paměť s náhodným přístupem). Typ paměti RAM se synchronním způsobem přenosu dat.
USB	Universal Serial Bus (univerzální sériová sběrnice). Univerzální sběrnice používána především k připojování periferií k počítači.
USB CDC	USB Communications Device Class. Přístrojová třída umožňující virtualizaci různých připojení přes USB.
USB OTG	USB On-The-Go. Specifikace umožnující zařízení plnit při komunikace jak roli master tak slave.
VGA	Video Graphics Array. Standart pro počítačovou zobrazovací techniku o maximálním rozlišení 640x480 px.
WVGA	Wide Video Graphics Array. Pojem používaný pro rozlišení se stejnou výškou jako VGA (480 px), ale větší šířkou.

Příloha B
Pinouty headerů použitých PCB

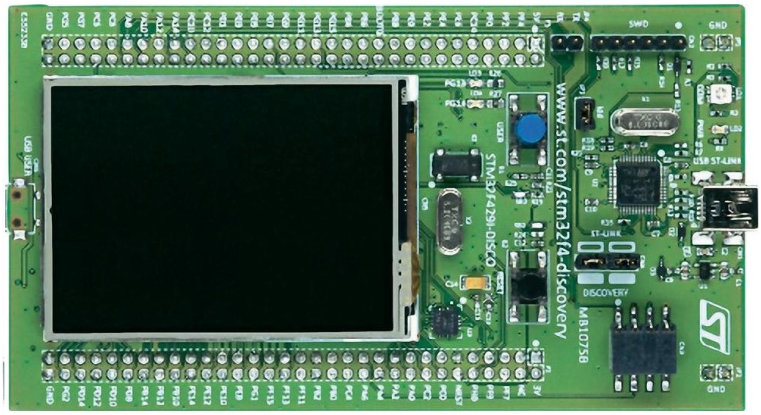


Obrázek B.1. Pinout propojovací desky mezi senzorem a kitem (Převzato z [7]).

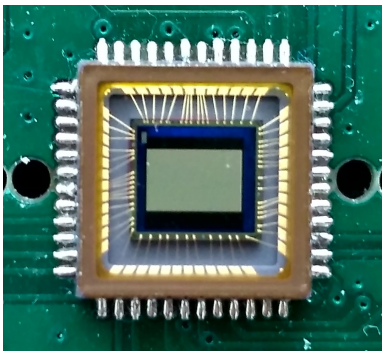


Obrázek B.2. CMOS header pinout (Převzato z [7]).

Příloha C
Fotodokumentace použitého HW



Obrázek C.3. STM32F429i-Discovery.



Obrázek C.4. CMOS senzor Aptina MT9V034.

Obrázek C.5. Porovnání použitých objektivů.

Obrázek C.6. Kompletní sestava kamery.

Příloha D
Obsah přiloženého CD

- `cameraFW` — FW pro vývojový kit
  - `sources` — zdrojové kódy
    - `Inc` — hlavičkové soubory .h
    - `Project` — soubory projektu μVision Keil 4
    - `Src` — zdrojové soubory .c
  - `cameraFW.hex` — hex soubor pro nahrání do vývojového kitu
- `data` — naměřená data
  - `osc` — časování signálů CMOS senzoru měřené na osciloskopu
  - `DCmotor.zip` — data naměřená na DC motoru
  - `shaker.zip` — data naměřená na shakeru
- `PCappliaction` — aplikace pro nadřazené PC
  - `DetectionApp` — zdrojové kódy
  - `DetectionApp.exe` — spustitelný soubor aplikace
  - `DetectionApp.sln` — Microsoft Visual Studio Solution file (project file)
- `thesis` — text práce
  - `sources` — zdrojová forma práce ve formátu LaTeX
    - `obrazky` — obrázky použité v rámci práce
  - `BP_Vodsedalek_Jakub_2017.pdf` — text práce ve formátu PDF
- `readme.txt` — stručný popis obsahu CD