VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ENERGETICKÝ ÚSTAV
ENERGY INSTITUTE

VYHODNOCENÍ VELIKOSTI A RYCHLOSTI KAPEK Z VYSOKORYCHLOSTNÍHO ZÁZNAMU
DETERMINATION OF DROPLET VELOCITY AND SIZE FROM HIGH-SPEED IMAGING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Jan Juroška

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Milan Malý, Ph.D.

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Energetický ústav
Student:	**Jan Juroška**
Studijní program:	Strojírenství
Studijní obor:	Základy strojního inženýrství
Vedoucí práce:	**Ing. Milan Malý, Ph.D.**
Akademický rok:	2020/21

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

Vyhodnocení velikosti a rychlosti kapek z vysokorychlostního záznamu

Stručná charakteristika problematiky úkolu:

Velikost a rychlost kapek ve spreji patří mezi základní charakteristiky určující kvalitu atomizace. Určení těchto parametrů z obrazového záznamu má zásadní výhody ve spolehlivé detekci největších a nesférických kapek, s nimiž mají problémy běžně používané optické metody jako např. Fázový Dopplerovský Anemometr (FDA). Cílem této práce je vyhodnocení vysokorychlostních záznamů kapek ve spreji a následné vyhodnocení velikosti a rychlosti kapek. Výsledky budou porovnány s daty z FDA.

Cíle bakalářské práce:

- rešerše relevantních publikací ke zpracování obrazu a detekci struktur
- předzpracování obrazu (vylepšení kontrastu, odečet pozadí, odstranění šumu atd.)
- vyhodnocení rychlosti kapek v Matlabu pomocí vlastního nebo volně dostupného kódu
- vyhodnocení velikosti kapek v Matlabu nebo ručně
- analýza vlivu zpracování obrazu na výsledek a odhad nejistoty detekce
- porovnání výsledků s daty z FDA

Seznam doporučené literatury:

ŠONKA, M., HLAVÁČ, V. & BOYLE, R., 2015. Image processing, analysis, and machine vision Fourth edition., CT: Cengage Learning

RUSS, J.C., 2011, The image processing handbook 6th ed., CRC Press

LEFEBVRE, A. H., AND MCDONELL, V. G., 2017, Atomization and sprays, CRC press

SZELISKI, R., 2010. Computer Vision: Algorithms and Applications, Springer

BSA Flow Software user guide v6.5, Skovlunde, 2017

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/21

V Brně, dne

L. S.

doc. Ing. Jiří Pospíšil, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Assignment Bachelor's Thesis

Institut:	Energy Institute
Student:	**Jan Juroška**
Degree programm:	Engineering
Branch:	Fundamentals of Mechanical Engineering
Supervisor:	**Ing. Milan Malý, Ph.D.**
Academic year:	2020/21

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Bachelor's Thesis:

Determination of droplet velocity and size from high–speed imaging

Recommended bibliography:

ŠONKA, M., HLAVÁČ, V. & BOYLE, R., 2015. Image processing, analysis, and machine vision Fourth edition., CT: Cengage Learning

RUSS, J.C., 2011, The image processing handbook 6th ed., CRC Press

LEFEBVRE, A. H., AND MCDONELL, V. G., 2017, Atomization and sprays, CRC press

SZELISKI, R., 2010. Computer Vision: Algorithms and Applications, Springer

BSA Flow Software user guide v6.5, Skovlunde, 2017

Deadline for submission Bachelor's Thesis is given by the Schedule of the Academic year 2020/21

In Brno,

L. S.

doc. Ing. Jiří Pospíšil, Ph.D.
Director of the Institute

doc. Ing. Jaroslav Katolický, Ph.D.
FME dean

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/2021.

V Brně dne

L.S.

prof. Ing. Josef Štětina, Ph.D.	doc. Ing. Jaroslav Katolický, Ph.D.
ředitel ústavu	děkan fakulty

Abstrakt

V rámci této práce byl vytvořen postup na získání rychlosti a velikosti kapek z obrazového záznamu pomocí počítačového vidění. Rychlost je získána pomocí metody *Particle Image Velocimetry* s použitím PIVlabu. Velikost a excentricita je získána pomocí vlastního programu v matlabu. Teoretické základy potřebné pro vytvoření programu jsou popsány v teoretické části. V praktické části ověřuji přesnost mojí metody a následně ji aplikuji na reálný sprej.

Klíčová slova

Počítačové vidění, zpracování obrazu, PIV, Particle Image Velocimetry, LoG, laplacián gaussiánu, laboratoř sprejů.

Abstract

In this bachelor thesis there was created a method of getting velocity and size of droplets form a video using computer vision. Velocity is obtained via PIVlab toolbox in matlab, wich uses Particle Image Velocimetry. Size is obtained via matlab code. Theoretical basis on wich this code was written is included in the theoretical part of this thesis. In practical part of this thesis, accuracy of this method was evaluated and the method was applied on a video of a real spray droplet flow.

Keywords

Computer vision, picture analysis, PIV, Particle Image Velocimetry, LoG, Laplacian of Gaussian, Spray Laboratory.

Bibliografická citace

Citace tištěné práce:

JUROŠKA, Jan. Vyhodnocení velikosti a rychlosti kapek z vysokorychlostního záznamu. Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/132158. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Energetický ústav. Vedoucí práce Milan Malý.

Citace elektronického zdroje:

JUROŠKA, Jan. Vyhodnocení velikosti a rychlosti kapek z vysokorychlostního záznamu [online]. Brno, 2021 [cit. 2021-05-12]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/132158. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Energetický ústav. Vedoucí práce Milan Malý.

Čestné prohlášení

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením

Ing. Milana Malého, Ph. D, s použitím informačních zdrojů uvedených v seznamu.

V brně dne

Jan Juroška

Brno 2021

Poděkování

Děkuji pánům prof. Ing. Janu Jedelskému, Ph.D, Ing. Milanu Malému, Ph.D a Ing. Danielu Adámkovi, z nichž jeden mi umožnil se na tuto metaforickou plavbu nalodit, druhý mi poskytl mapu a cíl mé cesty a třetí mi jako maják osvětlil všechna úskalí.

Brno 2021

Obsah

1 Úvod

Cílem této bakalářské práce je především navrhnout a vytvořit v prostředí Matlabu způsob, kterým bude možné analyzovat rychlost a velikost kapek na videozáznamu, tuto analýzu v praxi provést a posoudit její přesnost.

Stanovení těchto vlastností je u kapek důležité pro jejich aplikaci. Např. pro lepší spalování paliva jsou vhodné malé kapky, zatímco pro zemědělské aplikace jsou vhodné větší kapky, které se nedají snadno odfouknout větrem.

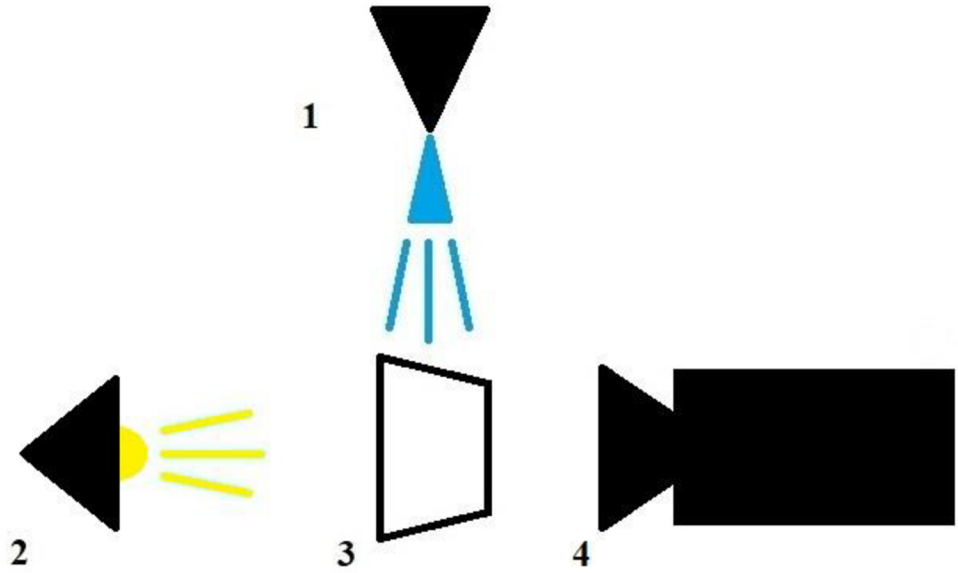
V rámci rešerše se budu zabývat vymezením počítačového vidění a zpracování obrazu, a také problémy, které při počítačovém zpracování obrazu nastávají. Přesněji popíšu problém, který budu řešit. Dále prozkoumám konkrétní přístupy, funkce a nástroje použitelné pro jeho řešení.

Praktická část mé práce spočívá v napsání programu v Matlabu, který bude mít za úkol ze série obrazů získat velikost a excentricitu kapek, a využití nástroje PIVlab k získání rychlosti kapek. Nakonec budu zkoumat přesnost mojí metody v závislosti na nastavení programu.

Metoda bude v laboratoři sprejů VUT FSI sloužit jako doplnění Fázové Dopplerovské Anemometrie, která není použitelná pro neprůhledné a nesférické kapky.

2 Analýza problému

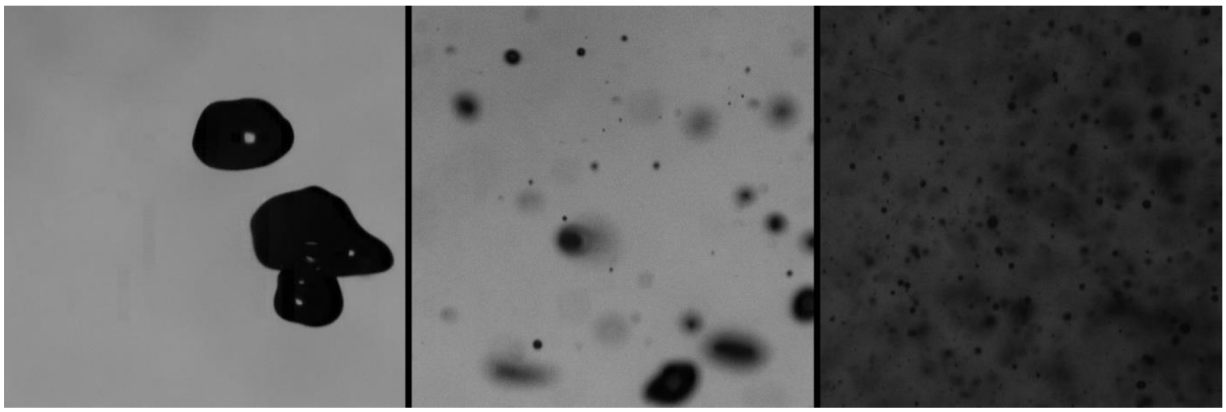
Získání obrazu probíhá tak, že tryska vstřikuje do prostoru kapičky, tyto kapičky jsou poté v určité vzdálenosti snímány kamerou a z druhé strany osvětlovány (viz obrázek 1).



obrázek 1: schéma snímání obrazu: tryska (1), světlo (2), zachycovaná scéna (3), kamera (4)

Z podoby záznamu (obrázek 2) lze získat mnoho apriori informací o obraze, které mi poslouží při navrhování programu.

Obrázky jsou v odstínech šedé. Na záznamu lze předpokládat různý počet kapek. Kapky mají různé velikosti, ale mívají charakteristické tvarové rysy. Může se vyskytovat i velmi silný šum. Dále vidíme že mnoho kapek není v hloubce ostrosti. Toto by mohl být značný problém, protože jejich průměr na obrázku neodpovídá jejich reálnému průměru (viz 0 Ztráta informací při převodu obrazu z 3D do 2D).



obrázek 2: příklady vzhledu záznamu

Tuto úlohu budu řešit v Matlabu. Matlab nabízí širokou škálu funkcí, knihoven a *toolboxů,* věnujících se počítačovému vidění. Na prototypování podobného programu je tedy vhodný.

Oproti jiným programovacím jazykům je ale výrazně pomalejší. Pokud tedy bude v budoucnu zájem o zrychlení programu, jako první krok bych doporučoval jeho přepis do jiného jazyka.

3 Počítačové vidění a zpracování obrazu

Počítačové vidění je mezioborová vědní disciplína, která se zabývá porozuměním obrazu nebo videozáznamu pomocí počítačů. Přesněji se zabývá zachycením obrazu a jeho zařazením do určitého předem zavedeného modelu světa [1]. V praxi se využívá především k získávání informací ze záznamu, od kontroly kvality výrobků [2] nebo automatického vytváření 3D modelů [3] po sledování populací hmyzu [4].

Zpracování obrazu se dělí na vyšší a nižší úroveň.

Na nižší úrovni zpracování jde o zachycení obrazu, jeho digitalizaci, úpravu do podoby, ve které ho budeme zpracovávat (předzpracování), a vymezení částí obrazu o které máme zájem (segmentace).

Na vyšší úrovni zpracování se provádí popis jednotlivých oblastí zájmu a další práce s těmito daty. Hranice mezi těmito úlohami, stejně jako mezi vyšší a nižší úrovní zpracování, nejsou jasně dané. V různých případech se mohou překrývat, probíhat zároveň nebo může být jedna operace zařaditelná do více kategorií. Různá literatura nám nabízí různé pohledy na tuto problematiku [1].

3.1 Nižší úroveň zpracování obrazu

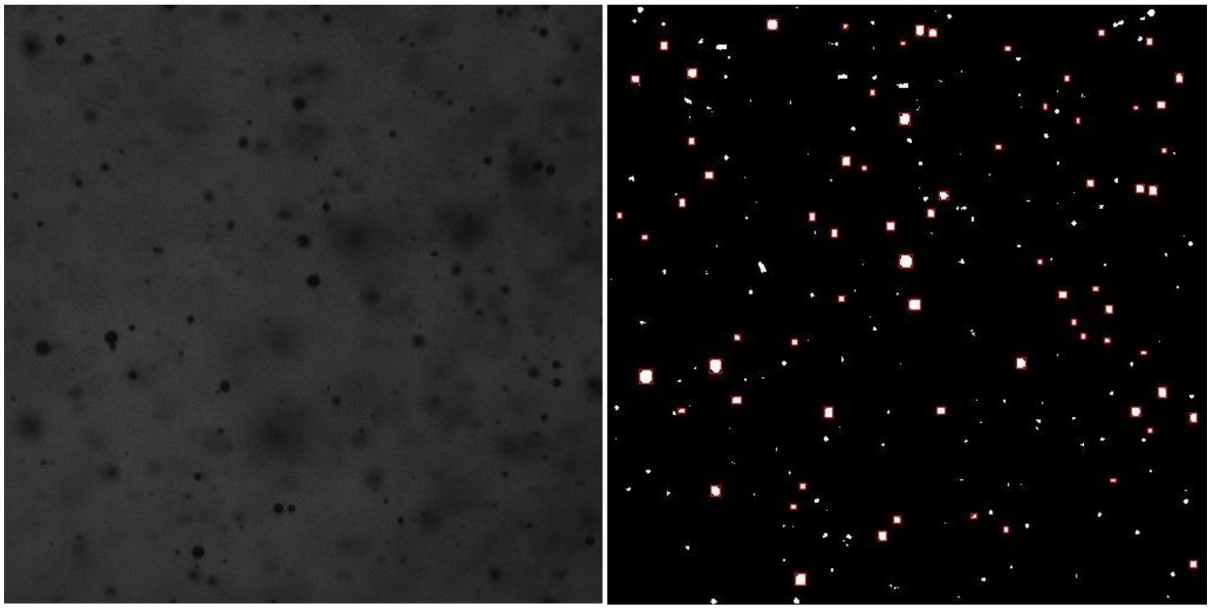
Vstupem na této úrovni zpracování je okolí, tak jak je zobrazeno pomocí kamery nebo fotoaparátu. Tento obraz je následně digitalizován, tj. převeden na obrazovou funkci nebo funkce. Ve většině případů se jedná o 2D obraz, tedy má obrazová funkce formu $f(x, y)$, kde souřadnice x, y udávají polohu pixelu. Je ale možné vytvořit také 3D obrazovou funkci, běžně s pomocí více kamer. Funkce potom každému pixelu přiřadí informace, typicky světlost nebo barvu [1].

3.1.1 Způsoby popisu obrazu

Funkce popisující obraz se dají dělit na spojité, diskrétní a digitální. Spojitá je funkce se spojitým oborem hodnot i definičním oborem. Pokud má funkce diskrétní obor hodnot a spojitý definiční obor, považujeme ji za diskrétní. Digitální funkcí označujeme takovou, která má diskrétní obor hodnot i definiční obor. Běžné zařízení (např. v mobilním telefonu) je schopno zachytit analogový obraz, reprezentovat jej spojitou funkcí a ihned digitalizovat [1].

3.1.2 Předzpracování

Předzpracování se zabývá převedením obrazu do takové formy, ve které ho bude možné zpracovávat na vyšší úrovni zpracování. Vstupem i výstupem je zde obrazová funkce. Typicky se jedná o úpravy obrazu, jako jsou odstranění šumu, zvýšení kontrastu, detekce hran apod.. Předzpracování upravuje informace, které jsou v obrazu obsaženy a může tedy být zdrojem zkreslení nebo ztráty dat, je tedy nutné u každé jeho funkce zvážit, zdali nepozměňuje i ty informace, které jsou měřeny [1]. Obrázek před a po předzpracování můžete vidět na obrázek 3.



obrázek 3: Příklady obrázku před a po předzpracování, červené čtverce okolo některých kapek byly dodány až v pozdější fázi a do předzpracování nespadají

3.1.3 Segmentace

V rámci segmentace jsou vyčleněny oblasti obrázku, na kterých bude vyšší úroveň zpracování provedena. Toto vede jednak v menšímu objemu zpracovaných dat, jednak k jednoduššímu procesu zpracování [1].

3.2 Vyšší úroveň zpracování obrazu

Na vyšší úrovni zpracování je obraz převeden na požadovaná data. Vstupem je obraz, výstup může být různý podle potřeby uživatele. Může se jednat o číslo, sérii písmen nebo třeba část obrazu. V dnešní době se v této oblasti široce využívá strojového učení, umělé inteligence a podobných prostředků. Stále se ale zcela běžně využívá klasických programů.

Zpracování na této úrovni typicky probíhá tak, že je nejdříve stanoven určitý model reality, se kterým je poté obraz porovnáván. Např. při rozpoznávání ručně psaných písmen by modelem byla databáze vzhledů jednotlivých písmen, jednotlivá písmena textu by pak byla s těmito modely porovnávána, dokud by nebyla nalezena nejpravděpodobnější shoda [1].

3.3 Problémy a překážky počítačového vidění

Počítačové vidění se snaží napodobit lidské vnímání obrazových vjemů. Tento úkol je velmi obtížný a při jeho řešení narážíme na množství překážek. V následujícím výčtu uvádím některé z nich [1].

Ztráta informací při převodu obrazu z 3D do 2D

Tento problém má více rovin, některé z nich jsou zcela stejné u lidí i počítačů. Člověk např. potřebuje dvě oči, aby byl schopen vizuálně vnímat vzdálenost (i když je schopen ji vnímat v rámci kontextu obrazu). Stejně pokud nemáme informaci o vzdálenosti předmětu, jen o jeho vzhledu, je velmi těžké posoudit jeho skutečnou velikost. Na toto téma bylo vytvořeno mnoho optických iluzí, které pomohou člověku uvědomit si tyto nedokonalosti.

V mém případě je ztráta informací způsobena nejen převodem koule z 3D na obrazec do 2D, ale také tím, že osvětlení je prakticky bodové.

Interpretace obsahu obrazu

Tento problém řeší člověk zcela bezmyšlenkovitě. Jsme například schopni bez problému určit židli, a to bez ohledu na její technické provedení. V případě počítače je třeba mít velkou databázi židlí, podle které by mohl posoudit, zda daný objekt je, nebo není židle. Nikdy však nebudeme schopni vytvořit databázi všech existujících židlí. Obecně tedy můžeme říct, že analýza obrazu může proběhnout jen tehdy, pokud umí program daný objekt zařadit.

Šum

Zkreslení v nějaké formě ovlivňuje všechna měření. Je třeba použít různé matematické metody, jako je např. teorie pravděpodobnosti, abychom šum odstranili. Použitím těchto metod ale nutně zkreslujeme obraz samotný. V některých případech můžeme zcela změnit některé pixely obrazu, v jiných snížit kontrast obrazu nebo ho naopak zvýšit.

Nepřesnosti zachycení obrazu

Světlost nebo i barvy získaného obrazu nejsou závislé pouze na objektech, které jsou snímány. Fyzika zachycení obrazu je relativně složitá disciplína a přijímané světlo může být zkresleno geometrií povrchu nebo třeba jeho schopností odrážet a pohlcovat světlo. V některých aplikacích může být také světlo upraveno po cestě k objektivu. Dalším zdrojem zkreslení může být zařízení, kterým je obraz snímán, např. vady objektivu.

Z těchto důvodů se v některých případech upouští od klasického fyzikálního zachycení obrazu a nahrazuje se jinými metodami [1].

Lokální zachycení obrazu

Obraz nám poskytne pouze lokální vhled na potenciální širší vzorek dat. Často je třeba globálního pohledu na problém. Díky tomu v některých případech nejsme schopni uvést přesné řešení problému, pouze jeho odhad.

4 Nižší úroveň zpracování

V rámci nižší úrovně zpracování se budu zabývat předzpracováním a segmentací obrazu. Obrázky mám už vytvořené, jejich zachycení a zápis do digitální podoby tedy není předmětem mojí práce.

4.1 Předzpracování

V rámci předzpracování si kladu za cíl vytvořit obraz, na kterém budou jasně patrné hranice kapek, a to tak, aby co nejvíce odpovídaly realitě. Také je vhodné, aby se detekované hranice kapek odlišovaly od zbytku obrázku dost na to, aby bylo možné provést účinnou segmentaci. V ideálním případě by detekce hran měla proběhnout tak, aby nedetekovala rozmazané hranice kapek.

4.1.1 Metody předzpracování

Funkce používané na předzpracování se dají obecně rozdělit na globální a lokální. Globální předzpracování provede buď stejnou operaci na všech pixelech, nebo bude podoba operace záviset na pozici pixelu v obrázku. Lokální funkce provádějí operace v závislosti na pixelech v okolí zpracovávaného pixelu.

Toto lokální předzpracování je realizováno z velké části pomocí masek. Maska je matice, která každému pixelu v nějakém okolí přiřadí váhu a na základě vážených hodnot pixelů provádí danou operaci. Ve své práci budu masky označovat písmenem h.

Pro představu uvedu masku používanou k vyhlazení obrázku

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \qquad 1$$

Zde se hodnota pixelu v upraveném obrázku vypočítá jako průměrná hodnota pixelů v jeho 3×3 okolí v původním obrázku.

Tato verze masky ale silně stírá okraje, proto se někdy využívá maska, která klade větší důraz na původní pixel nebo jeho čtyřokolí [1].

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad 2$$

4.1.2 Detekce hran

Hledání okrajů regionů obrázku se nazývá detekce hran. Spočívá v nalezení výrazných změn v obrazové funkci a jejich označení. Tento proces využívá toho, že objekty mívají po své ploše velmi podobné schopnosti odrážet a pohlcovat světlo. Obecně lze říct, že hrana se nachází tam, kde se jasová funkce ostře mění [1].

Při analýze obrazu se dá použít směrová detekce hran, která zároveň určuje normálu a někdy i tečnu hrany, nebo hledá pouze hrany s určitým směrem normály. V našem případě ale postačí nesměrová detekce hran, která detekuje pouze přítomnost hrany obecně.

Pro detekci výrazné změny funkce se přímo nabízí využití derivací. Lze použít např. hledání lokálních maxim první derivace. Velmi efektivním a často používaným způsobem se ukázala detekce hran pomocí protnutí nuly druhou derivací. Při ostré změně hodnoty intenzity její první derivace prudce vzroste a v místě jejího maxima druhá derivace nulu ostře protne, viz obrázek 4. Tento bod je mnohem lépe a přesněji určitelný pomocí protnutí nuly druhou derivací, než pomocí maxima první derivace [5].

obrázek 4: funkce hrany (červená), její první (modrá) a druhá (zelená) derivace

Pro nesměrové zjištění hodnoty druhé derivace lze použít Laplaceův operátor, definovaný jako

$$\Delta = \nabla^2 = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2}, \qquad 2$$

kde při detekci hran je n počet dimenzí funkce obrázku a x_i je označení jednotlivých souřadnic těchto dimenzí.

Operátor je v základní formě neovlivněný rotací obrázku, aproximuje se konvoluční sumou. Někdy se zvýrazňuje vliv určitých pixelů v závislosti na tom, jaké okraje chceme detekovat. Jako příklad uvádím několik masek, schopných tímto způsobem detekovat okraje 2D obrazu. U masek h_3 a h_4 je kvůli tomu ztracena netečnost k rotaci [1].

$$h_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad h_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \qquad 3$$

$$h_3 = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix}, \qquad h_4 = \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}, \qquad 4$$

Tento operátor, nazývaný laplacián, je poměrně citlivý na šum. Spolehlivější užití operátoru dosáhneme, pokud obrázek nejdříve vyhladíme. Toto se může ukázat jako problematická úloha, protože vyhlazováním obrázku se snižuje strmost funkce intenzity obrázku.

Při výběru vyhlazovacího filtru by měla být splněna dvě kritéria [5]. Zaprvé by měl být dostatečně hladký a na dostatečně malé oblasti, aby se co nejvíce omezil možný počet prudkých změn funkce v rámci masky filtru. Zadruhé by měl být na tak velké oblasti, aby zohlednil dostatečně velké okolí vyšetřovaného pixelu.

Jinými slovy by filtr měl být tak malý, aby vyhlazoval pouze jednu významnou změnu v intenzitě obrázku, ale zároveň dost velký na to, aby vyhlazení mohlo proběhnout podle dostatečně vzdálených pixelů.

Tyto dva požadavky jsou v přímém rozporu, ale lze je optimalizovat s pomocí Gaussova rozdělení. 2D Gaussův vyhlazovací operátor, nazývaný gaussián je definován jako

$$G(x,y) = e^{\frac{-(x^2+y^2)}{2\sigma^2}}, \tag{5}$$

kde x, y jsou souřadnice a σ je charakteristika rozdělení, určující velikost okolí, na kterém filtr pracuje.

Zdá se tedy vhodné zkombinovat tyto dva operátory a získat tak jediný operátor, který funkci zároveň vyhladí a zároveň detekuje její hrany. Výsledkem této kombinace je tzv. laplacián gaussiánu, běžně označovaný jako LoG (Laplacian of Gaussian) [1]

$$\nabla^2[G(x,y,\sigma) * f(x,y)] = \nabla^2[G(x,y,\sigma)] * f(x,y). \tag{6}$$

Díky linearitě operátorů můžeme obměnit pořadí konvoluce a diferenciace

$$\nabla^2 G(x,y,\sigma) = \frac{1}{\sigma^2}\left(\frac{x^2+y^2}{\sigma^2} - 2\right) e^{\frac{-(x^2+y^2)}{2\sigma^2}}. \tag{7}$$

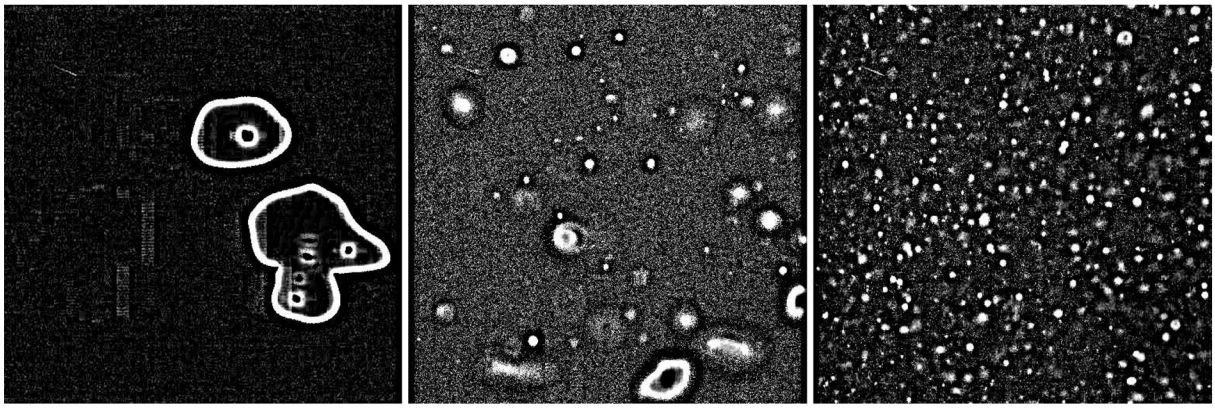
Derivaci Gaussova filtru lze pak spočítat analyticky

$$h(x,y) = c\left(\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right) e^{\frac{-(x^2+y^2)}{2\sigma^2}}. \tag{8}$$

Maska poté vzniká přidáním normalizačního koeficientu c, který normalizuje součet všech prvků matice na nulu.

Výhodou této metody je její variabilita a spolehlivost. Nevýhodou je velikost použitých masek, např. pro $\sigma = 4$ je pro spolehlivý výpočet potřeba masky široké 40 pixelů [1].

obrázek 5 ukazuje výsledek při použití filtru o šířce 17 pixelů a $\sigma = 0{,}25$.



obrázek 5: příklad obrázků po použití LoG

Vidíme, že i přes šum se mi podařilo kapky s použitím LoG spolehlivě detekovat. Na obrázcích je sice silné zrnění, v okolí hranic zaostřených kapek ale ne. V okolí nezaostřených kapek toto zrnění přítomno je, to by mohlo později přispět k jejich eliminaci. Tento filtr obecně shledávám jako vhodný.

4.1.3 Odstranění šumu po detekci hran

Po detekci hran vznikl na obrázku silný šum tam, kde se v okolí nenachází žádná hrana. Tento šum je třeba odstranit tak, aby nebyly zpětně rozostřeny už detekované hrany. Není tedy možné opět použít klasický vyhlazovací filtr.

Přístupů k odstranění šumu bez rozostření hran je více. Já se budu zajímat o dva z nich.

První se nazývá vyhlazování rotující maskou. Hledá nejhomogennější okolí pixelu a novou hodnotu pixelu vypočítá jako průměr hodnot v tomto okolí.

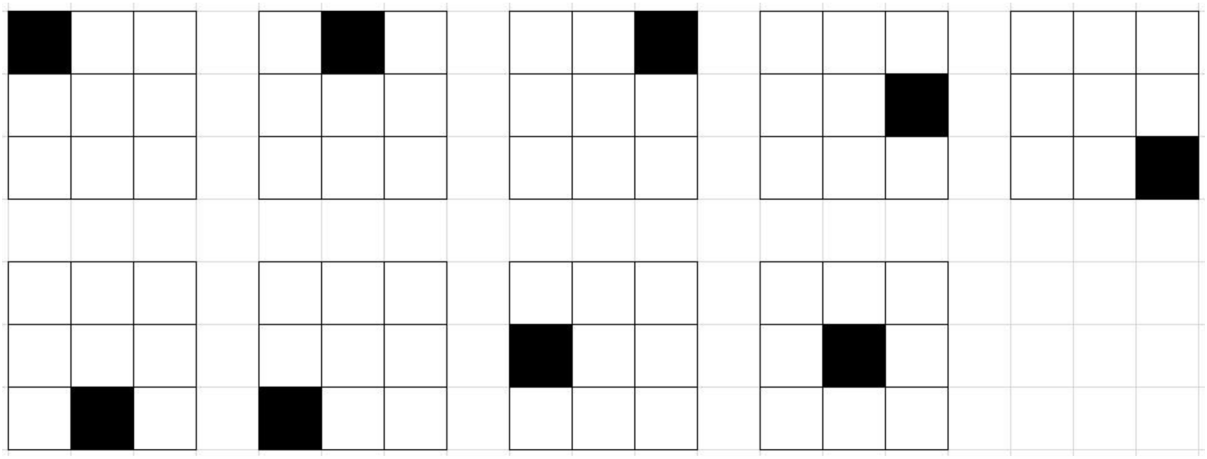
Druhým je mediánový filtr, který nemá charakter vyhlazování jasové funkce, místo toho přiřadí pixelu nejčastější hodnotu pixelů v jeho okolí.

Metoda vyhlazování rotující maskou

Metoda vyhlazení rotující maskou je nelineární metoda zajímavá především tím, že její použití hrany nerozostřuje, ale zostřuje [6]. Tato metoda se zakládá na výpočtu rozptylu jasu pixelů v masce. Rozptyl jasu σ^2 je definován jako

$$\sigma^2 = \frac{1}{n} \sum_{(i,j)\in R} \left(g(i,j) - \frac{1}{n} \sum_{(i,j)\in R} g(i,j) \right)^2 . \qquad 9$$

Spočítá se ve všech maskách o daném tvaru v nějakém okolí. Jako příklad uvedu možné 3×3 masky, které pokrývají 5×5 okolí, viz obrázek 6 [1].



obrázek 6: Možné 3x3 masky pro výpočet rozptylu

Jas pixelu se pak určí jako průměr jasů pixelů v masce s nejnižším rozptylem.

Mediánový filtr

Tento filtr je poměrně jednoduchý. Spočívá v určení masky a nalezení nejčastější hodnoty v masce. Pixelu je pak přiřazena tato nejčastější hodnota [1].

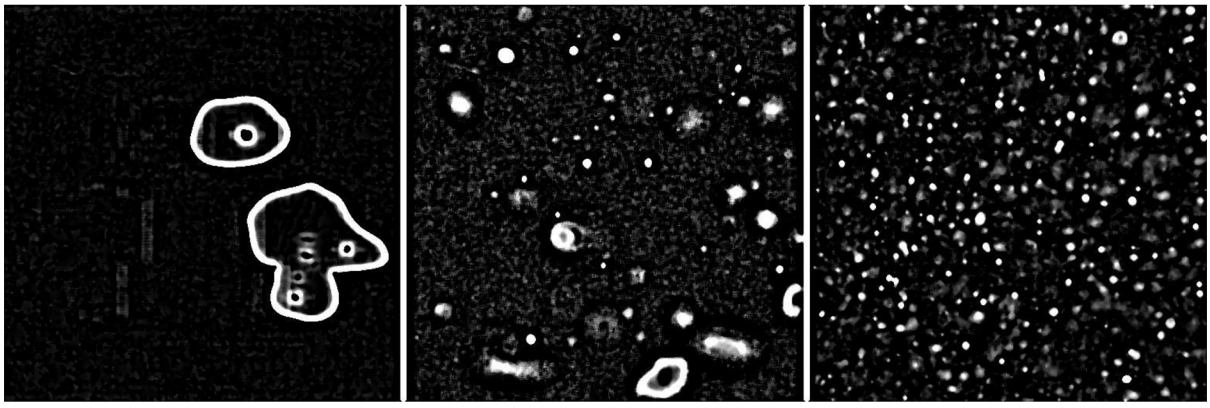
Tato metoda se zdá jako vhodná pro odstranění zrnitého šumu, který vznikl po použití LoG. Může ale změnit tvar kapek, a to zejména u kapek o velikosti menší, než je velikost masky. Velikost této změny je menší, použijeme-li menší masku.

Volba metody

Rotující maska obraz vyhladí, takže po každé části šumu vždy zbude stopa. Její největší výhodou je, že vznikne hladký obraz s výraznými hranami. Tato vlastnost pro mě ale není příliš důležitá. Hrany jsou již dostatečně výrazné a hladkost obrázku není prioritou.

Mediánový filtr má radikálnější přístup k potlačování zrnění. Drtivá většina bílých pixelů zrnění je obklopena černými pixely. Nedá se ale jednoznačně říct, že drtivá většina černých pixelů je obklopena bílými. Předpokládám tedy, že zrnění bude účinněji potlačeno. Ze zrnění také pravděpodobně vzniknou oblasti s menším jasem než hrany, ale vyšším jasem než zbytek pozadí. Další malou výhodou mediánového filtru je výpočetní nenáročnost.

Po vyzkoušení obou metod shledávám jako vhodnější mediánový filtr (obrázek 7), který v kombinaci s prahováním (4.2.1 Prahování) dosahuje dobrých výsledků.



obrázek 7: obrázky po použití mediánového filtru s velikostí masky 5x5

4.2 Segmentace

Segmentace je obzvláště důležitý krok v procesu zpracování obrazu. Jeho vstupem je předzpracovaný obraz, a jeho cílem je obraz rozdělit na části, které mají silnou korelaci s reálnými objekty, které nás zajímají.

Dělí se na úplnou segmentaci, jejímž výsledkem jsou disjunktní regiony, z nichž v každém je jeden reálný objekt, a částečnou segmentaci, při které nemusí regiony zcela odpovídat reálným objektům a nemusí být disjunktní [1].

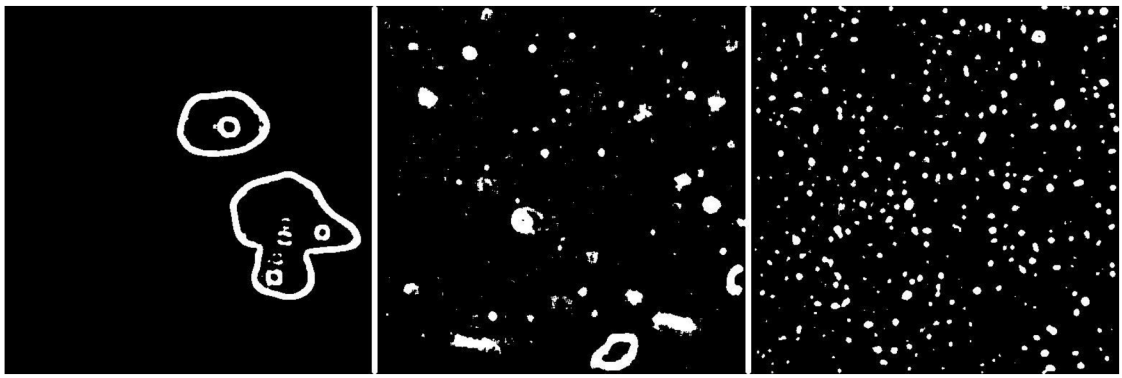
Obvykle je k tomuto třeba použít složitějších postupů. V některých případech, jako je ten, kterým se zabývám já, si lze vystačit i s jednoduššími funkcemi.

V rámci segmentace si kladu za cíl jasně oddělit pozadí obrázku od kapek, jednoznačně označit každou jednotlivou kapku a tak připravit obrázek na následné zpracování.

4.2.1 Prahování

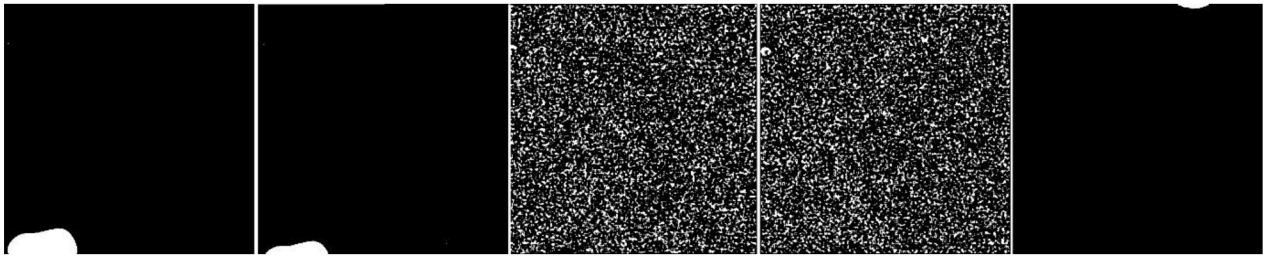
Prahování je jeden z nejstarších přístupů k segmentaci. Je nenáročné a u jednodušších aplikací velmi účinné.

Často mají objekty zájmu na obrázku stejnou schopnost odrážet nebo pohlcovat světlo. V takových případech lze určit jistou konstantu jasu, práh, pod, resp. nad kterou se objekty nachází. Prahování potom všechny pixely, které se přes tento práh nedostanou, označí jako pozadí a jejich hodnotu přepíše na 0, všechny ostatní pixely označí jako objekty a jejich hodnotu nastaví na 1. Obrázky po prahování můžeme vidět na obrázek 8.



obrázek 8: Obrázky po prahování

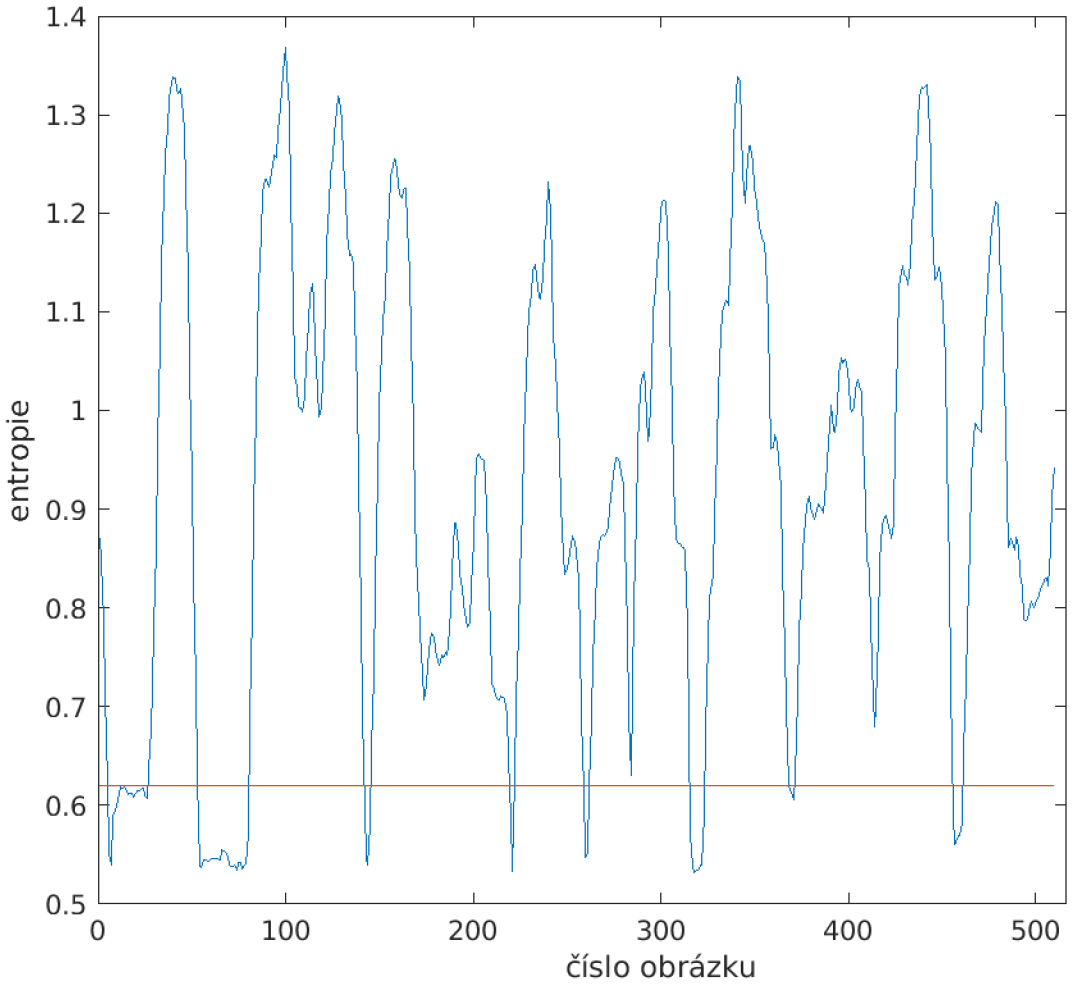
Prahování funguje problematicky, pokud je na obrázku příliš málo nebo žádné hrany. Pak není možné určit práh podle úrovně jasu hran a funkce ho určí mezi úrovněmi pozadí, které vznikly po aplikaci mediánového filtru. Výsledkem je zcela zničený obrázek (obrázek 9).



obrázek 9: série snímků kamery, kapky padají směrem dolů, je patrné zničení obrázků bez kapek

Tento problém lze řešit pomocí entropie. Entropii obrázku si lze intuitivně představit jako míru nejistoty, s jakou jsme schopni odhadnout hodnotu pixelu. V mém případě lze tedy určit konstantu entropie, nad kterou se nevyskytují žádné obrázky zničené tímto způsobem.

Tuto konstantu lze určit jako nejvyšší číslo z množiny entropií zničených obrázků (obrázek 10). Účinnost metody jsem testoval na několika sadách souborů, každý obsahoval zhruba 500 obrázků. Ze všech obrázků pod konstantou entropie ani v jednom případě počet nezničených nepřesáhl počet deseti, zatímco počty odfiltrovaných obrázků se pohybovaly v desítkách.



obrázek 10: Entropie jednotlivých obrázků (modrá) a limit entropie (oranžová)

4.2.2 Označení jednotlivých regionů

Posledním úkolem segmentace je regiony označit. Tento proces probíhá tak, že pixely každého unikátního spojitého regionu jsou přepsány na stejné unikátní číslo.

5 Vyšší úroveň zpracování

V rámci této úrovně zpracování chci získat velikost a excentricitu relevantních kapek a také jejich průměrnou velikost a excentricitu.

Pro přehlednost jsem se rozhodl ponechat původní názvy proměnných a funkcí, tak jak jsou uvedeny v Matlabu.

5.1 Získání požadovaných parametrů

V Matlabu je k podobným účelům určena funkce *regionprops*. Tato funkce vyšetří každý objekt a zjistí u něj požadované parametry. Parametrů, které je schopna zjistit, je více. V mém kódu budu používat pět z nich.

5.1.1 Area

Parametr *Area* nám udává, kolik pixelů celkově objekt tvoří. Vzhledem k tomu, že naše objekty jsou zatím pouze hranice kapek, nikoliv kapky samotné, tento parametr neodpovídá velikosti kapky.

Velikost kapky bychom ovšem získali, pokud bychom nejprve vnitřek každé kapky vyplnili. K tomu se dá použít funkce *imfill*, která vyplní všechny části černobílého obrázku, ze kterých se nelze dostat na jeho okraj, aniž bychom přešli z černé na bílou.

Tento parametr má formát čísla.

5.1.2 BoundingBox

Tento parametr vrací nejmenší čtverec, který lze okolo regionu opsat a zároveň má stěny rovnoběžné s okraji obrázku. Aby bylo zřejmé, na kterém pixelu se přesně hranice nachází, nejsou souřadnice udány v celých pixelech, ale vždy v polovinách. Je tedy např. udáno, že severní hranice je 14.5 pixelů, to znamená, že ve vzdálenosti 14 pixelů od severního okraje se region nenachází, ale ve vzdálenosti 15 pixelů už ano.

Tento parametr má formát čtyř čísel, kdy první, resp. druhé udává souřadnici x, resp. y levého horního rohu čtverce. Třetí, resp. čtvrté číslo udává šířku, resp. výšku čtverce v pixelech. Počátek souřadnicového systému je v levém horním rohu.

5.1.3 Centroid

Parametr *Centroid* vrací těžiště regionu. Tento parametr posloužil pouze k zaměření jednotlivých kapek při optimalizaci konstant kódu, na výstupu se nijak nepodílí. Rozhodl jsem se ho ale v kódu ponechat, aby bylo v budoucnu možno kalibrovat kód dle potřeby.

Má formát dvou čísel, které udávají souřadnice x a y.

5.1.4 Eccentricity

Tento parametr udává excentricitu regionu. Tato excentricita je vypočítána jako excentricita elipsy se stejným druhým obecným momentem, jako má region.

Má formát čísla z intervalu $\langle 0; 1 \rangle$, kde nula a jedna jsou degenerované stavy. Nula je ve skutečnosti kružnice, jedna je ve skutečnosti úsečka.

5.1.5 ConvexArea

Parametr *ConvexArea* určí velikost nejmenšího konvexního tvaru, který lze vyšetřovanému regionu opsat.

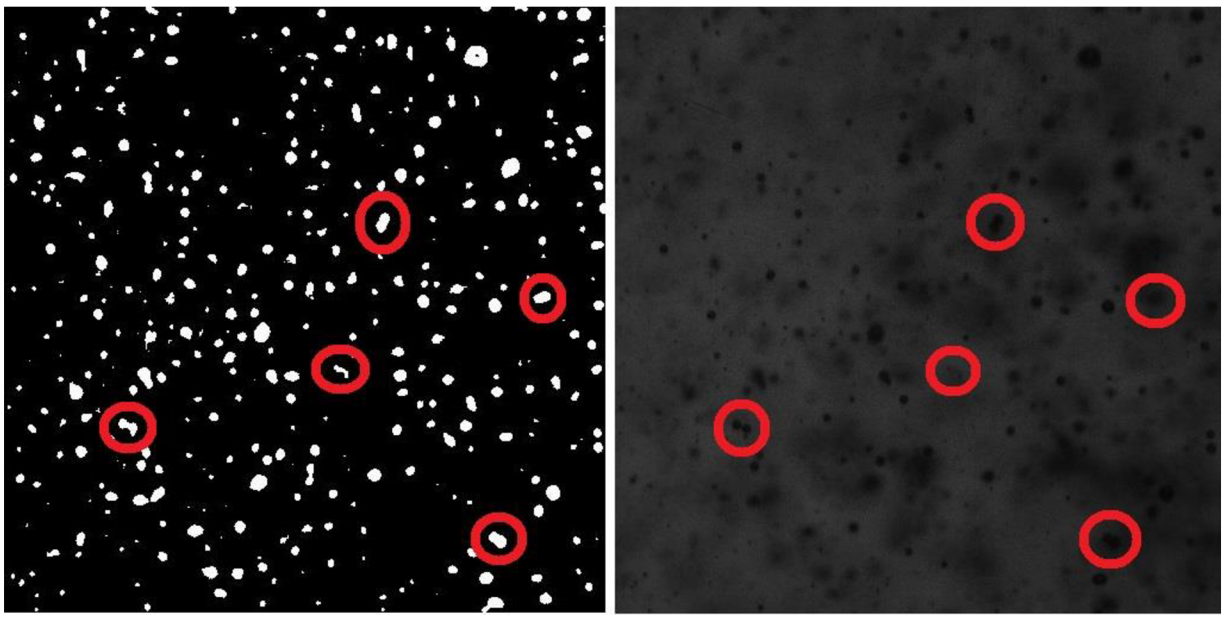
Tento parametr má formát čísla.

5.2 Odstranění regionů nesplňujících podmínku

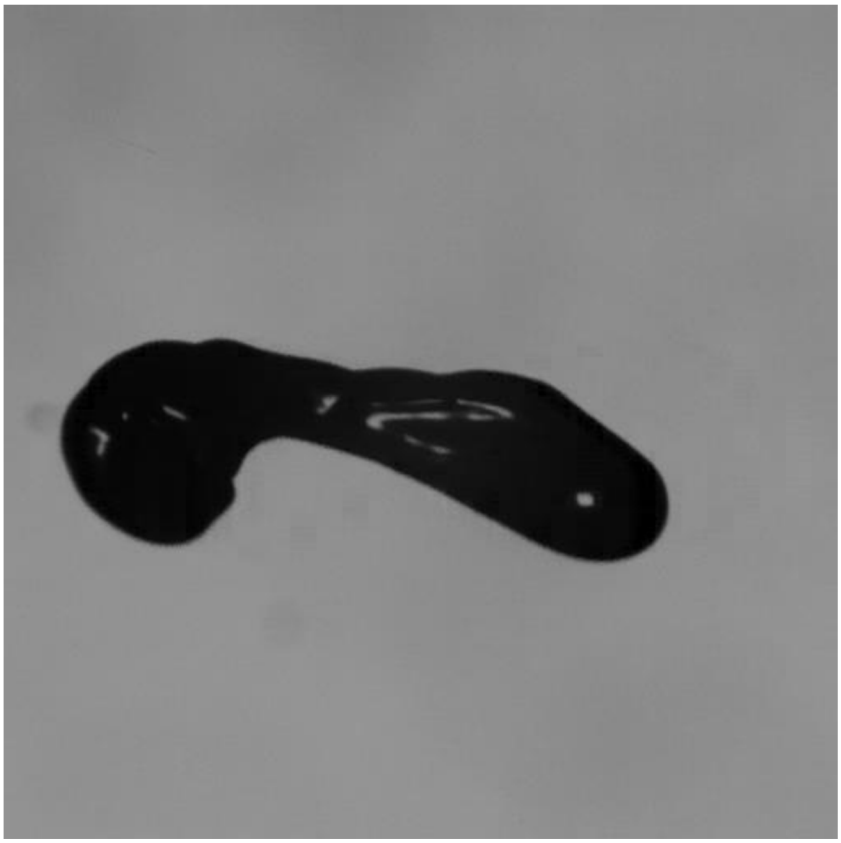
Ne všechny kapky byly dokonale segmentovány. Musím tedy vytvořit nástroj, který smaže informace o každé kapce, která nesplní nějakou podmínku. Takto by se měla eliminovat většina nepřesností v detekci.

5.2.1 Podmínka excentricity

První podmínka, kterou jsem stanovil, je maximální excentricita. Tato podmínka by měla eliminovat kapky, které se vzájemně překrývají, viz obrázek 11. Podmínka je zmírněna, popř. zcela odstraněna, pro větší kapky. U větších kapek se nedá obecně říct, že by měly vysokou excentricitu pouze v případě překrytí, viz obrázek 12.



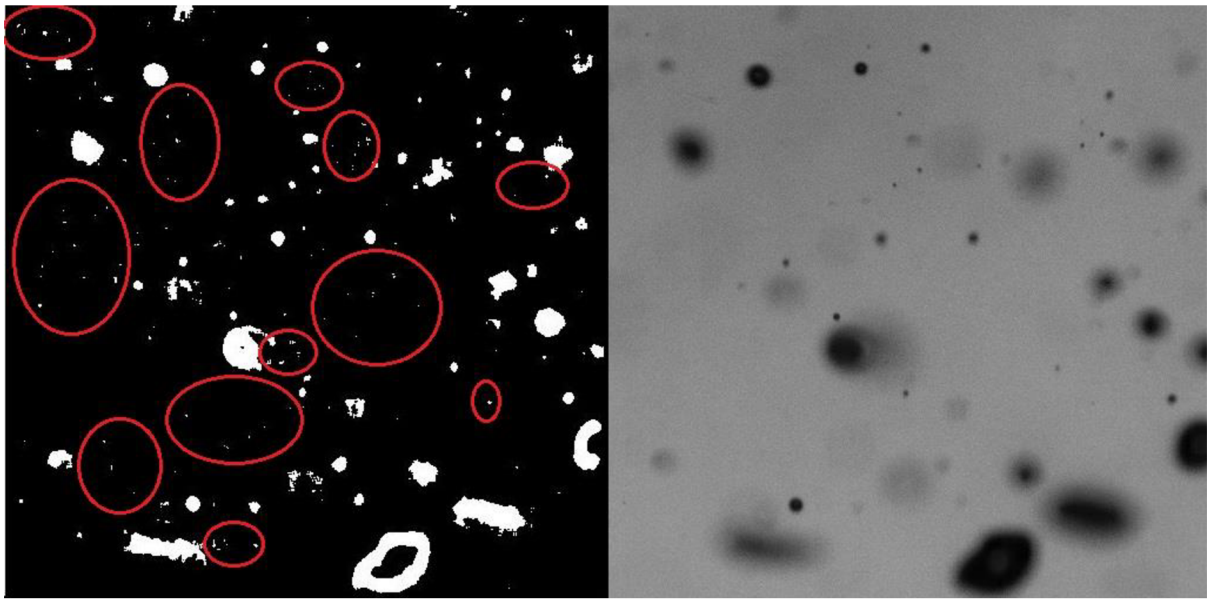
obrázek 11: Některé příklady vysoké excentricity u menších překrytých kapek, Vlevo je předzpracovaný obrázek, vpravo původní obrázek



obrázek 12: Příklad velké kapky s vysokou excentricitou

5.2.2 Podmínka velikosti

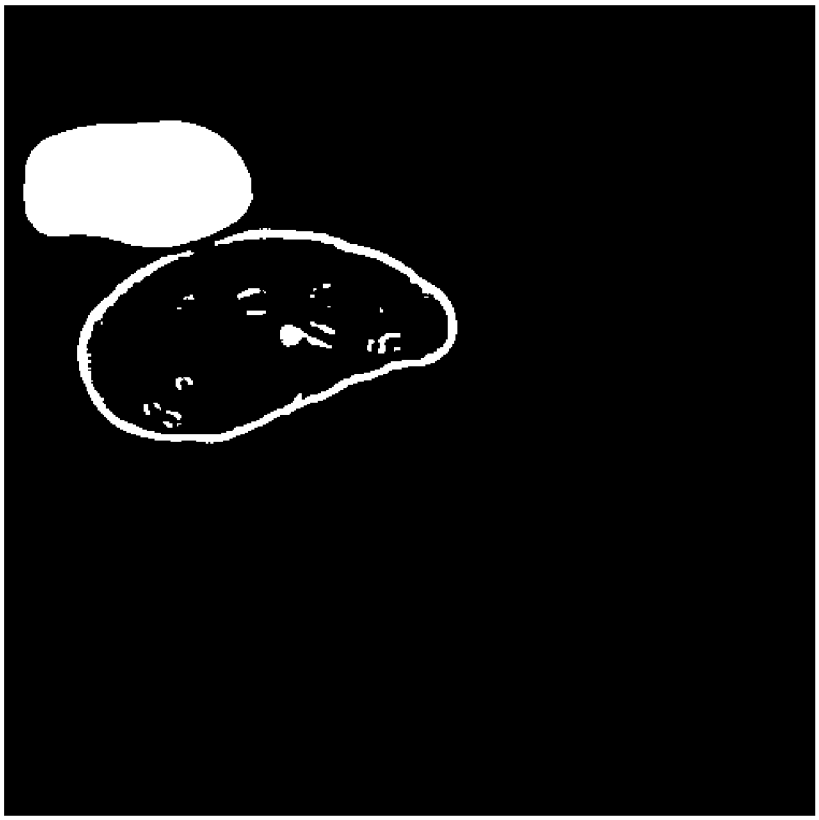
Další podmínkou je velikost regionu. Na obrázku je stále přítomen šum v podobě malých pixelů (viz obrázek 13), tento šum zde vzniká např. jako pozůstatek rozmazaných kapek nebo čmouh na objektivu. Pro přesnější měření je tedy lepší ignorovat regiony o malých velikostech.



obrázek 13: příklady šumu odstíněného podmínkou velikosti (vlevo), vidíme že na původním obrázku (vpravo) mnoho z nich vůbec není rozlišitelných

5.2.3 Podmínka vyplnění

Dosud jsem v tomto dokumentu používal převážně obrázky, na kterých nebylo provedeno vyplnění kapek pomocí funkce *imfill*. Po aplikaci funkce ale vidíme, že u neuzavřených kapek se tato funkce neprovede, viz obrázek 14.

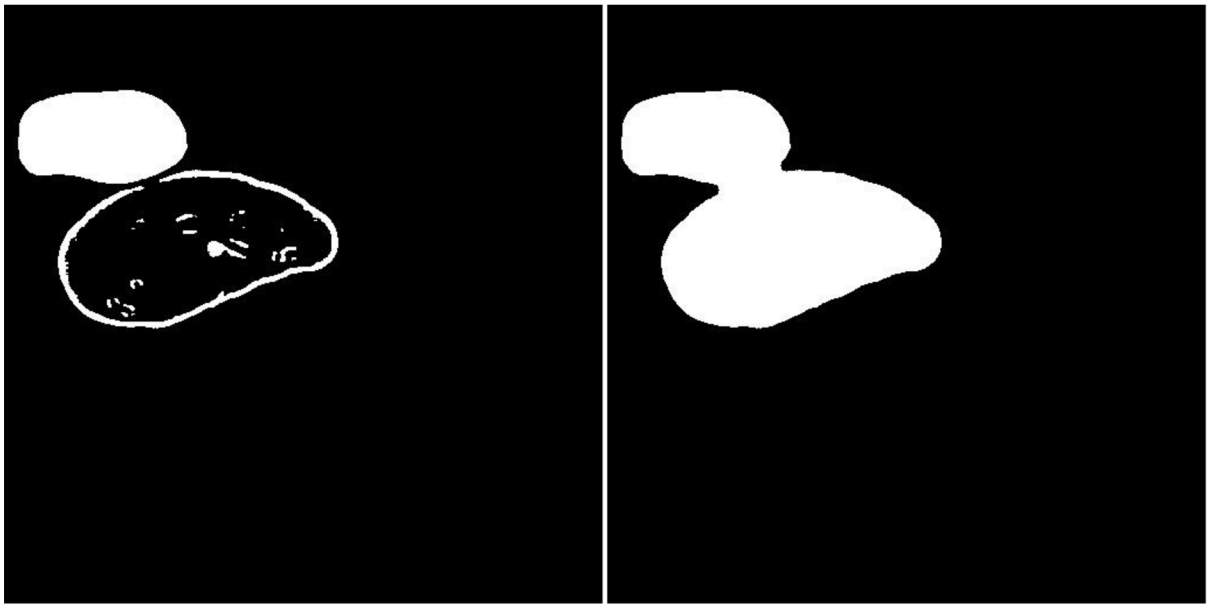


obrázek 14: Příklad neuzavřené kapky, která nebyla vyplněna

Tento problém se dá řešit dvěma způsoby. Buď můžu ignorovat neuzavřené kapky, nebo se je budu snažit uměle uzavřít.

K umělému spojování blízkých hran se běžně používá morfologická operace uzavírání. Tato operace spočívá ve dvou po sobě následujících úkonech, dilataci a erozi [1]. Problém této operace je, že upravuje tvar kapek a zaplňuje nejen mezery v jednotlivých hranách, ale také mezi kapkami blízko u sebe, viz obrázek 15.

Tento přístup tedy shledávám jako nevhodný.



obrázek 15: vyplnění kapek bez (vlevo) a s (vpravo) předchozím uzavřením obrázku

Budu tedy volit ignorování neuzavřených kapek. Tuto podmínku formuluji jako maximální povolený podíl parametrů *Area* a *ConvexArea*.

ConvexArea udává povrch nejmenšího opsaného konvexního tělesa, a ten by se neměl v případě vyplněné kapky výrazně lišit od parametru *Area*. V případě nevyplněné kapky bude ale parametr *Area* řádově menší.

5.2.4 Podmínka obsažení v nevyplněné kapce

Můžeme si všimnout (obrázek 15), že uvnitř nevyplněných kapek je spousta šumu, vzniklého různou schopností pohlcovat a odrážet světlo napříč kapkou.

Regiony tedy nejsou brány v úvahu, nachází-li se jejich *BoundingBox* uvnitř *BoundingBoxu* kapky, odstraněné na základě podmínky vyplnění.

5.2.5 Podmínka pozice

Kapky, které se nacházejí na okraji obrázku, s vysokou pravděpodobností nejsou celé (viz obrázek 16). Některé z nich jsou odstraněny na základě podmínky vyplnění, ale jiné, zejména malé, nejsou. Proto je vhodné neuvažovat kapky, jejichž *BoundingBox* je příliš blízko okraji obrázku.

obrázek 16: kapky na okraji které v obrázku nejsou celé

5.3 Určení průměrných hodnot a přepočítání na reálné jednotky

V rámci vyšší úrovně zpracování dále určuji průměrnou velikost a excentricitu napříč všemi kapkami které prošly přes podmínky. Poslední úloha je pak přepočet parametru *Area* z pixelů na jednotku plochy.

6 Struktura metody

Jako první krok by se měla určit rychlost. Rychlost je pro další analýzu důležitá, protože může nastat situace, kdy jednu kapku započítáme dvakrát a tím zkreslíme statistku. Program je proto nastaven tak, že bere v potaz každý x-tý obrázek. Pokud známe rychlost, můžeme pak bezpečně určit minimální počet obrázků, který by se měl po každém zpracování přeskočit. Je ovšem taky možné tento počet ze snímků odečíst „ručně“, pokud jsou kapky dobře rozlišitelné. Je důležité zohlednit, že kapky blízko okrajů obrázku nejsou brány v potaz, pokud tedy máme relativně velké kapky, může být lepší zvolit počet přeskočených snímků menší a tak zajistit detekci většího počtu kapek.
Rychlost jsem se ze záznamu rozhodl získat s pomocí PIVlabu [7][8][9]. PIVlab je *toolbox,* který byl původně vytvořen k analýze záznamu proudění vzduchu, pracuje na bázi metody zvané *Particle Image Velocimetry* (PIV) [7]. Postupně se z něj vyvinul užitečný nástroj pro získání vektorového pole rychlosti ze záznamu.

PIV nejdříve rozdělí obrázky na mnoho menších oken, pak srovnává dva po sobě následující obrázky a zjišťuje posunutí jednotlivých oken. Zadáme-li také časový rozdíl mezi snímky, jsme schopni dostat vektorové pole rychlostí pro jednotlivá okna.

Můj program pracuje za předpokladu, že uživatel bude upravovat konstanty jednotlivých funkcí, a bude ho tak možné dle potřeby kalibrovat. Drtivá většina použitých funkcí je totiž nastavitelná. Lze upravovat např. velikosti jednotlivých masek nebo podmínek, které musí region splnit. Ideální nastavení programu se potom odvíjí od podoby záznamu, typu spreje a požadavků na určení.

Program lze rozdělit na tři části.

V první části (obrázek 17) se načítá databáze obrázků a definuje se, na kterých obrázcích bude program pracovat. Určí se po kolika snímcích bude program přeskakovat, první snímek a poslední snímek. Pokud by měl program zpracovat snímek s vyšším pořadím, než má poslední, zastaví se.

```
close all;
clear all;
clc;

% načtení databáze-------------------------------------------------------------------------------------
% zadáte cestu podle vzoru, na konci se přečtou všechny .tif soubory, nepřečte to nic v podsložkách
allImages = imageDatastore('/MATLAB Drive/bakalařka/Data/d0.95-4_3bar_Z800/*.tif','LabelSource','foldernames');


Labels = countEachLabel(allImages);
totalNumberOfImages = Labels.Count;      % určuje celkový počet obrázků

skok = 10;                               % skok udává ob kolik obrázků bude program pracovat
firstPictureNumber = 1;                  % udává první zpracovaný obrázek
lastPictureNumber = 100;                 % lastPictureNumber udává po který obrázek se to bude zpracovávat
pictureNumber = firstPictureNumber;      % NEMĚNIT, první obrázek vždy nastavit přes firstPictureNumber
```

obrázek 17: první část kódu

V druhé části (obrázek 18) probíhá cyklus zpracování jednotlivých obrázků.

Nejdříve zjišťuji, jestli je obrázek nad limitem entropie, pokud ne, přechází program na další, dokud nenajde obrázek nad limitem entropie.

```
21      %načtení obrázku pokud na něm něco je
22      limitEntropie = 0.62;
23      pictureNumber = odstranPrazdneObrazky(pictureNumber, allImages, limitEntropie);
24      obrazek = readimage(allImages,pictureNumber);
```

```
222 function pictureNumber = odstranPrazdneObrazky(pictureNumber, allImages, limitEntropie)
223     obrazek = readimage(allImages,pictureNumber);
224     entropie=entropy(obrazek);
225     while limitEntropie > entropie
226         pictureNumber = pictureNumber+1;
227         obrazek = readimage(allImages,pictureNumber);
228         entropie=entropy(obrazek);
229     end
230 end
```

obrázek 18: kontrola entropie, horní část obrázku je přímo v kódu, dolní je psaná jako samostatná funkce

Následně provádí předzpracování a segmentaci, a to v pořadí LoG, mediánový filtr, prahování, vyplnění regionů, označení regionů (viz obrázek 19). Vyzkoušel jsem všechna možná uspořádání těchto funkcí a zdá se, že toto pořadí dosahuje nejlepších výsledků.

```
28      %detekce hran pomocí gausiánu laplaciánu
29      w=fspecial('log',[7 7], 0.05);
30      filtrovanyObrazek=imfilter(obrazek,w,'replicate');
31
32      %úprava mediánovým filtrem a převod na binárnní obraz
33      binarniObrazek = medfilt2(filtrovanyObrazek,[5 5]);
34      binarniObrazek = imbinarize(binarniObrazek);
35
36      %vyplnění regionů
37      binarniObrazek = imfill(binarniObrazek,'holes');
38
39      %segmentace-----------------------------------------------------
40      %označení jednotlivých regionů obrazu
41      [regionyObrazku, puvodniPocetRegionu] = bwlabel(binarniObrazek);
```

obrázek 19: předzpracování a segmentace

V následujícím kroku zjišťuji parametry regionů, následně pak kontroluji regiony na jednotlivé podmínky. Idea této kontroly je zřetelná z obrázek 20. Podmínka obsažení v nevyplněné kapce je složitější, protože musí být zacyklena v další podmínce, princip je ale stejný.

```
104        %tady upravuju jak male kapky to maximálně vezme, zadavam velikost plochy v pixelech
105        podminkaArea = 15;
106        %vymazání příliš malých kapek
107        odebraneRadkyArea = 0;
108        for i = 1:pocetRegionu
109           kontrolaPodminkyArea = stats(i - odebraneRadkyArea).Area;
110           if podminkaArea > kontrolaPodminkyArea
111               stats(i - odebraneRadkyArea)=[];
112               odebraneRadkyArea = odebraneRadkyArea + 1;
113           end
114        end
```

obrázek 20: kontrola podmínky Area

U kapek, které prošly všemi podmínkami, potom vykreslím jejich *BoundingBox*. Následně se obrázek uloží. I když tyto kroky nejsou z hlediska řešení problému nutné, rozhodl jsem se je udělat, abych mohl lépe posoudit, zda program funguje tak, jak jsem předpokládal.

Nakonec se data získaná z obrázku přepíšou do struktury *superStats*, kde se takto nashromáždí data ze všech obrázků.

V třetí části (obrázek 21) se vypočítají průměrné hodnoty excentricity a velikosti kapek. Velikost se přepočítá na jednotku plochy. Z výstupní struktury se vymažou data, která nejsou pro uživatele zajímavá (např. parametr *BoundingBox*), nakonec se tato struktura uloží jako tabulka.

```
197    %přepočítání plochy z pixelu na absolutní jednotku plochy
198    pxNamm = 1;
199    plochaPixelu = (1/pxNamm)^2;
200    for i = 1:superPocetRegionu
201        superStats(i).Area = superStats(i).Area*plochaPixelu;
202    end
203    %uložení průměrných hodnot do struktury
204    jmenaSloupcuSuperStats = fieldnames(superStats);
205    pocetSloupcuSuperStats = numel(jmenaSloupcuSuperStats);
206    vystupCisla = transpose(reshape(cell2mat(struct2cell(superStats)), pocetSloupcuSuperStats, []));
207    prumerArea = mean(vystupCisla(:,1));
208    prumerEcc = mean(vystupCisla(:,2));
209    structAverageArea.Average_Area = prumerArea;
210    structAverageEcc.Average_Eccentricity = prumerEcc;
211    superStats = struct('Area',{superStats.Area},'Eccentricity',{superStats.Eccentricity}, 'Average_Area',
212        {structAverageArea.Average_Area},'Average_Eccentricity',{structAverageEcc.Average_Eccentricity});
213    %uložení struktury superStats
214    writetable(struct2table(superStats), 'stats.xlsx')
```

obrázek 21: třetí část kódu, pro přehlednost je řádek 211 rozdělen a pokračuje na řádku 212

7 Testování metody

Metodu nejdříve otestuji na dvou záznamech, pro které byla data získána i jinou metodou, a to pomocí fázové dopplerovské anemometrie. Tyto výsledky potom porovnám se svými. Protože anemometrie získává výsledky ve formě průměru ekvivalentní sférické kapky, vždy přepočítám parametr *Area* na průměr kruhu s obsahem rovným tomuto parametru. Následně svou metodu aplikuji na záznam reálného spreje, na kterém jsou zachyceny relativně velké a nesférické kapky, nelze jej tedy spolehlivě měřit metodou anemometrie.

Fázová dopplerovská anemometrie (FDA) je neintrusivní bodová metoda měření rychlosti a velikosti jednotlivých kapek, procházejících měřícím objemem. Měřící objem je definován dvěma laserovými paprsky, které se pod definovaným úhlem kříží a interferují spolu. Kapka v měřícím objemu generuje signál, který zachycují fotodetektory v přijímací optice. Rychlost kapek je stanovena na základě frekvence signálu. Velikost kapek je určena na základě fázového posuvu mezi dvojicí fotodetektorů, jež jsou na měřící objem zaostřeny pod výškovým úhlem. Tato metoda je schopna spolehlivě měřit pouze sférické a průhledné kapky.

Měření pomocí anemometru jsem neprováděl, data mi byla dodána vedoucím.

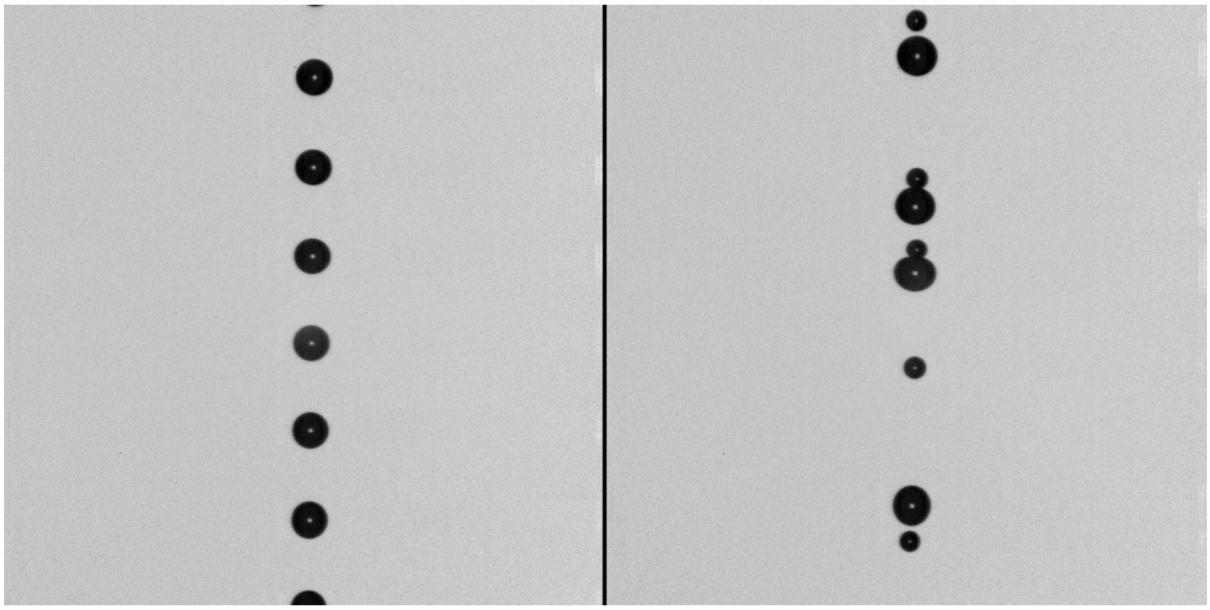
Pro pořízení záznamu byla použita kamera FASTCAM SA-Z (Photron, Japonsko). Sprej byl osvětlen pulsním LED světlem HPLS-36D18B (Lightspeed Technologies, USA). Pro obrázky, které jsem dosud v textu používal byla délka pulsu 400 ns s 1000 ns závěrkou, kamera snímala rychlostí 60 000 snímků za sekundu.

7.1 Monodisperzní atomizér

Jako první metodu otestuji na monodisperzním atomizéru. Tyto záběry (obrázek 22) byly zachyceny jako validační, měly by ověřit přesnost mojí metody. Jsou prakticky bez šumu. Kapičky jsou si velmi podobné, jsou malé a sférické, údaje z fázové dopplerovské anemometrie by tedy měly být poměrně přesné.

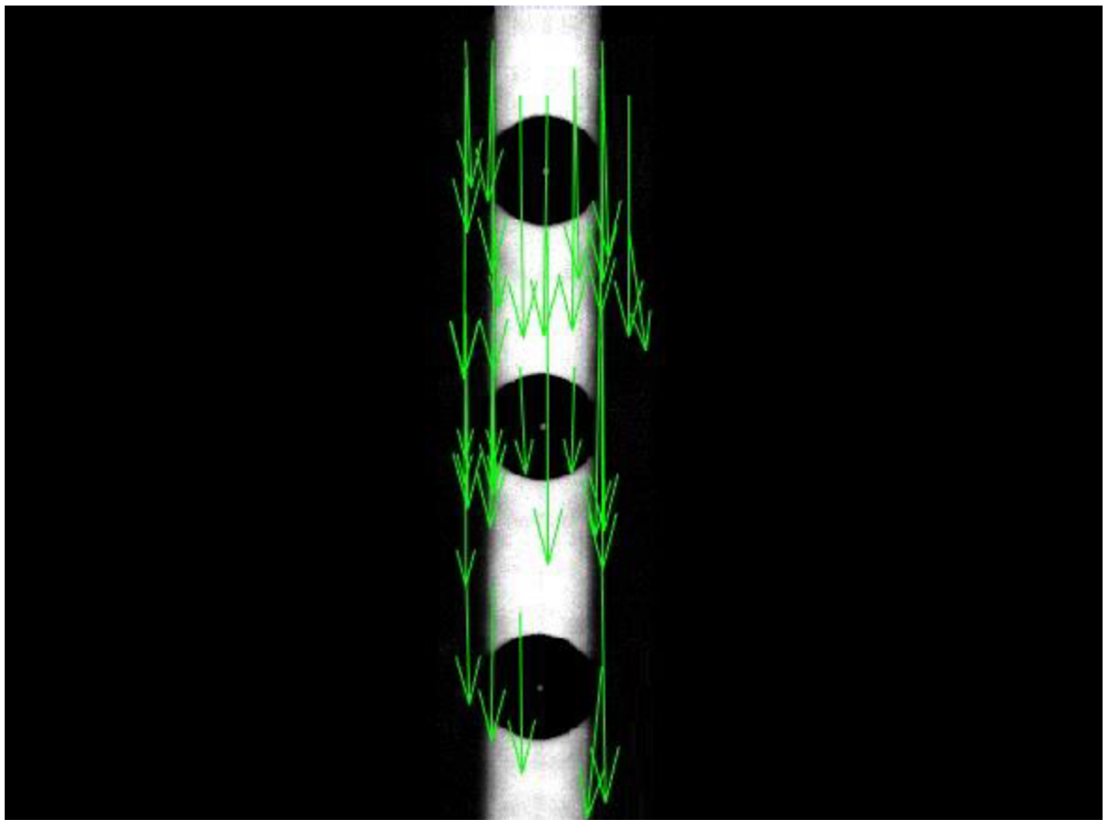
Mým cílem je mít výsledky obou metod podobné, ideálně by se měly metody navzájem doplňovat. Srovnání jsem provedl na dvou záznamech, na prvním jsou monodisperzní kapky, na druhém je dvouvelikostní rozložení. Srovnání mohlo být ovlivněno tím, že měření fázovou dopplerovskou anemometrií bylo provedeno pro 50 000 kapek, kdežto moje metoda byla provedena pouze na cca 150 kapkách.

Při pořízení záznamu byla délka impulsu světla 900 ns s 1 000 ns závěrkou, kamera snímala rychlostí 20 000 snímků za sekundu.



obrázek 22: ukázka testovacího záznamu, nalevo vidíme monodisperzní a napravo dvouvelikostní rozložení velikosti kapiček

Nejdříve jsem tedy v PIVlabu získal rychlost kapiček (viz obrázek 23). S pomocí referenčního obrazu jsem zjistil, že obrázek má asi 171 pixelů na milimetr. Při tomto měřítku se kapičky posunou o 75 pixelů za obrázek, tedy mají rychlost 8,77 metrů za sekundu. Počet obrázků, který by se měl přeskočit, aby nedošlo k dvojímu zachycení jedné kapky, je čtrnáct. Rychlost byla v obou případech stejná.



obrázek 23: výřez vektorového pole rychlosti kapek v PIVlabu

7.1.1 Jednovelikostní rozložení

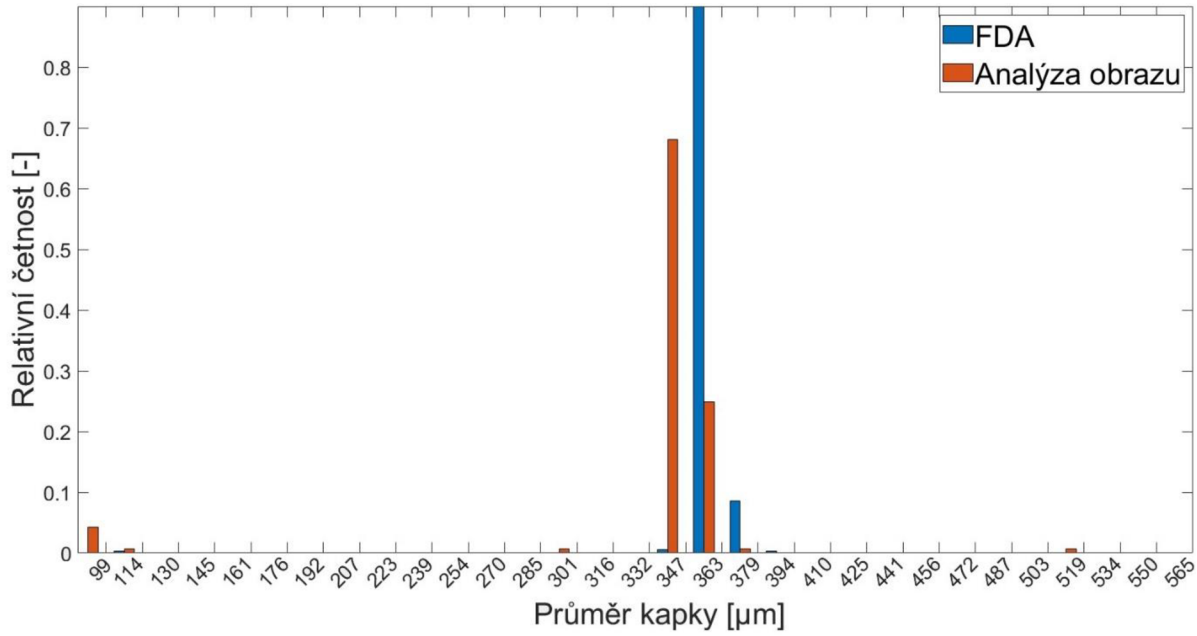
Měření bylo provedeno nejdříve při parametru LoG $\sigma = 0{,}6$, s maskou o velikosti 7×7, mediánový filtr měl masku o velikosti 5×5. Velikosti jednotlivých podmínek byly následující:

limitEntropie = 0,62
podminkaVyplneni = 0,7
podminkaArea = 60
podmínkaEccVelke = 0,9
podminkaKapkaArea = 50
podminkaBBox = 5

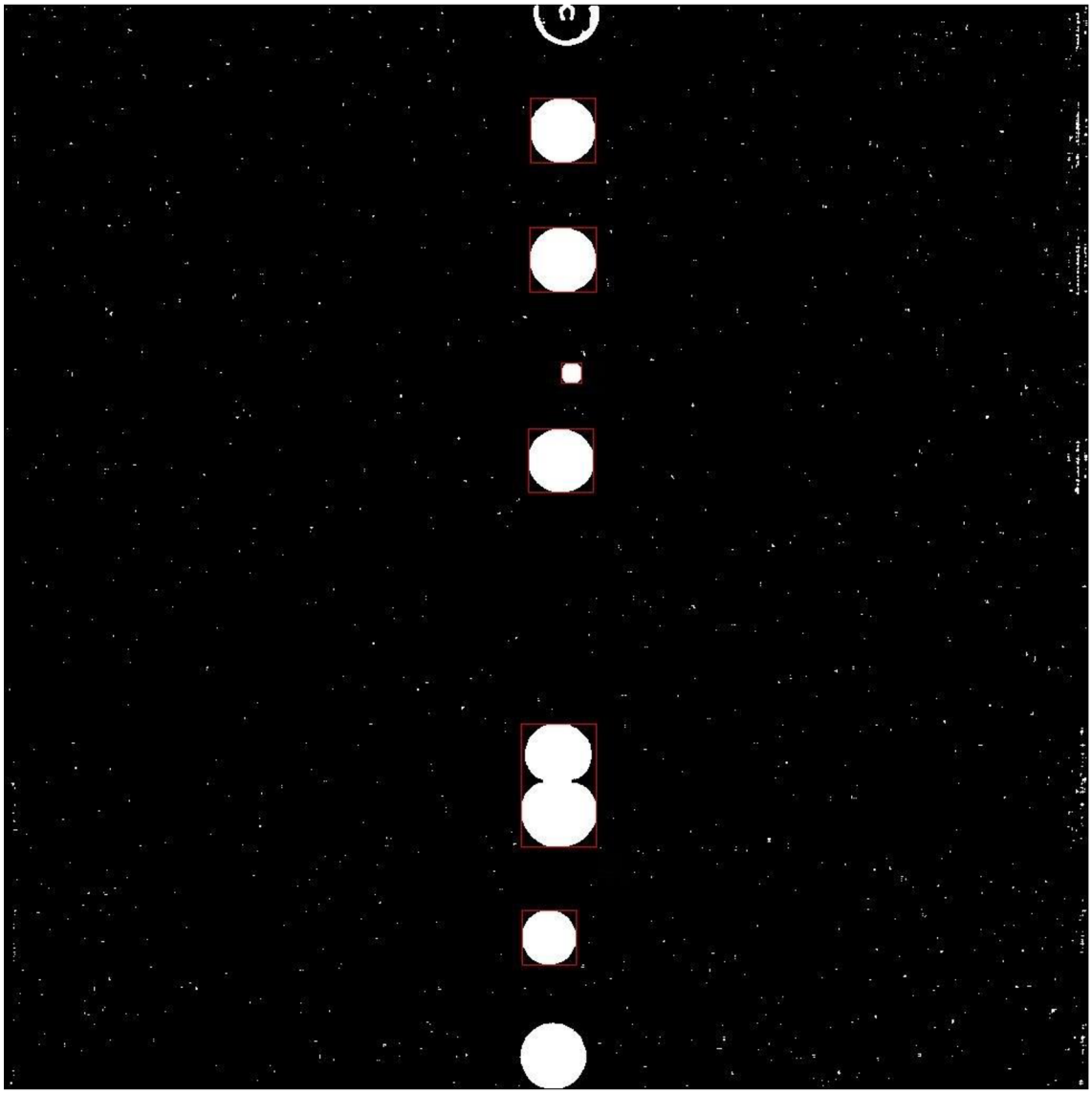
pozn.: podminkaEccMale zde není zmíněna, protože tak malé kapky jsou odfiltrovány podmínkou Area

Při tomto nastavení jsem změřil průměrnou hodnotu průměru kapky 354,3 µm, což je o 5,8% menší hodnota než 376 µm získaných pomocí fázové dopplerovské anemometrie. Z grafu relativních četností (obrázek 24) můžeme vidět, že toto bylo především způsobeno tím, že obrazová analýza detekovala kapičky o trochu menší.

Za zmínku také stojí jedna výrazně větší kapka (viz obrázek 25), detekovaná pomocí obrazové analýzy, toto jsou ve skutečnosti dvě kapky, které se v letu začaly spojovat, jedná se tedy o chybu, tato chyba je odstranitelná snížením podmínky *podmínkaEccVelke*. Jako další je patrný malý počet menších kapiček, které nebyly pomocí fázové dopplerovské anemometrie výrazně detekovány (viz obrázek 24 a obrázek 25).



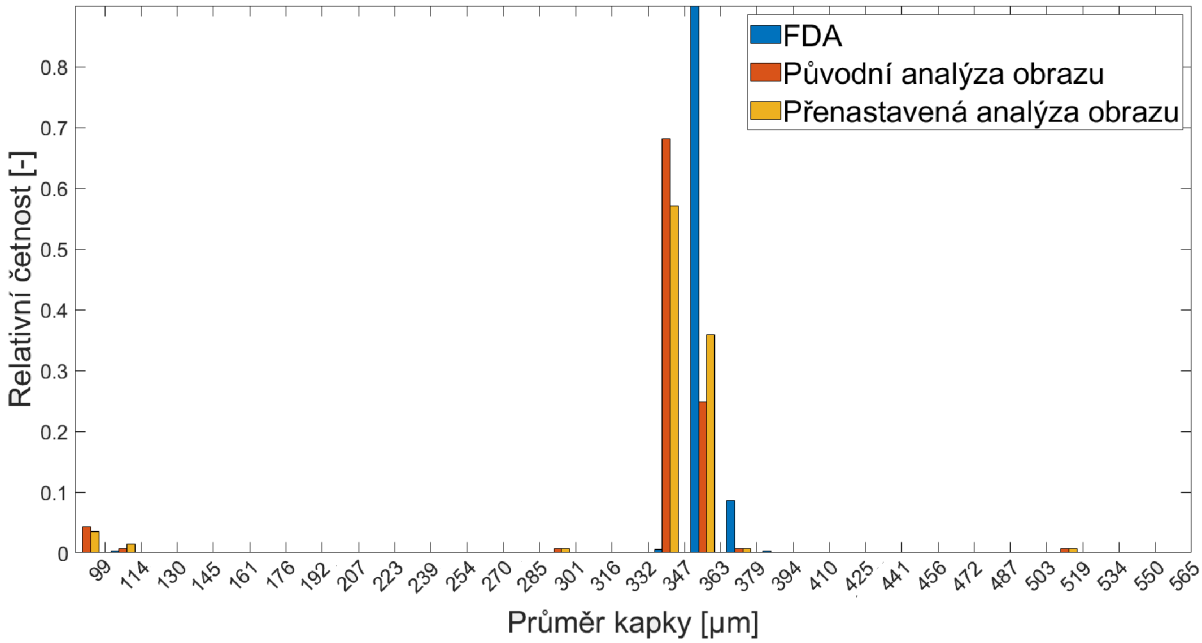
obrázek 24: histogram fázové dopplerovské anemometrie a prvního měření



obrázek 25: obrázek se spojenými kapkami, můžeme si zde také všimnout jedné z detekovaných menších kapiček, na obrázku jsou červeně vykresleny bounding boxy kapek, které prošly podmínkami

Test jsem provedl znovu s lehce pozměněným nastavením. Změněno bylo nastavení LoG, a to ze $\sigma = 0{,}6$ na $\sigma = 0{,}03$, v důsledku toho v obrázku vznikl výrazný šum, takže jsem zároveň zvýšil podmínku *Area* na $podminkaArea = 200$.

Při tomto nastavení jsem změřil průměrnou hodnotu velikosti kapky 356,1 µm, čímž jsem se k hodnotě získané pomocí fázové dopplerovské anemometrie přiblížil o asi 0,5%, změna tedy neměla výrazný vliv. Z histogramu (viz obrázek 26) můžeme vidět, jak se hmotnostní rozložení detekovaných kapek změnilo.



obrázek 26: histogram fázové dopplerovské anemometrie, prvního a druhého měření

Nejlepší představu o přesnosti metody dostaneme, pokud se do průměrné velikosti kapek nezapočítají malé kapičky, které fázová dopplerovská anemometrie neregistruje. Toho jsem docílil pomocí zvýšení podmínky *Area* na $podminkaArea = 1000$. Průměrná velikost je pak 364,1 µm, což je o 3,2% menší než velikost udávaná fázovou dopplerovskou anemometrií.

7.1.2 Dvouvelikostní rozložení

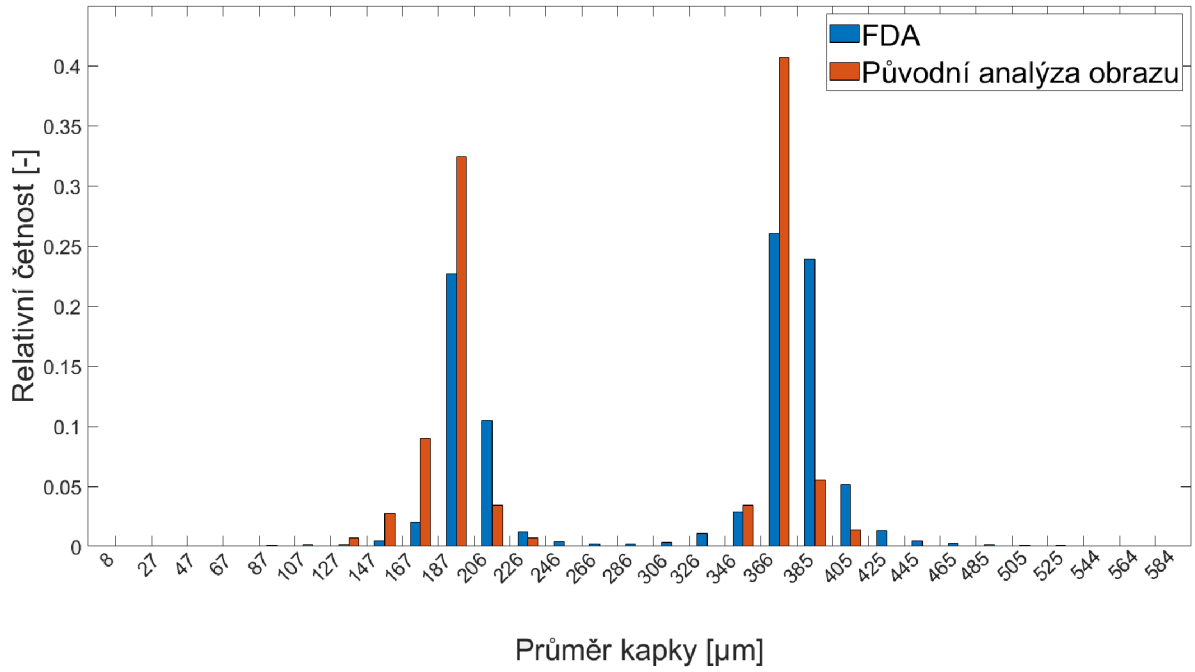
Měření jsem provedl s parametry, se kterými jsem u jednovelikostního rozložení provedl druhé měření. Jedinou výjimkou byl parametr *podmínkaEccVelke*. Protože se zde vyskytovalo mnoho sloučených a také pár excentrických kapek (viz obrázek 28), musel jsem ho snížit na 0,62. Ostatní zůstává stejné, tedy parametr LoG $\sigma = 0{,}03$, s maskou o velikosti 7×7, maska mediánového filtru o velikosti 5×5. Velikosti jednotlivých podmínek byly následující:

limitEntropie = 0,62
podminkaVyplneni = 0,7
podminkaArea = 200
podmínkaEccVelke =0,62
podminkaKapkaArea =50
podminkaBBox = 5

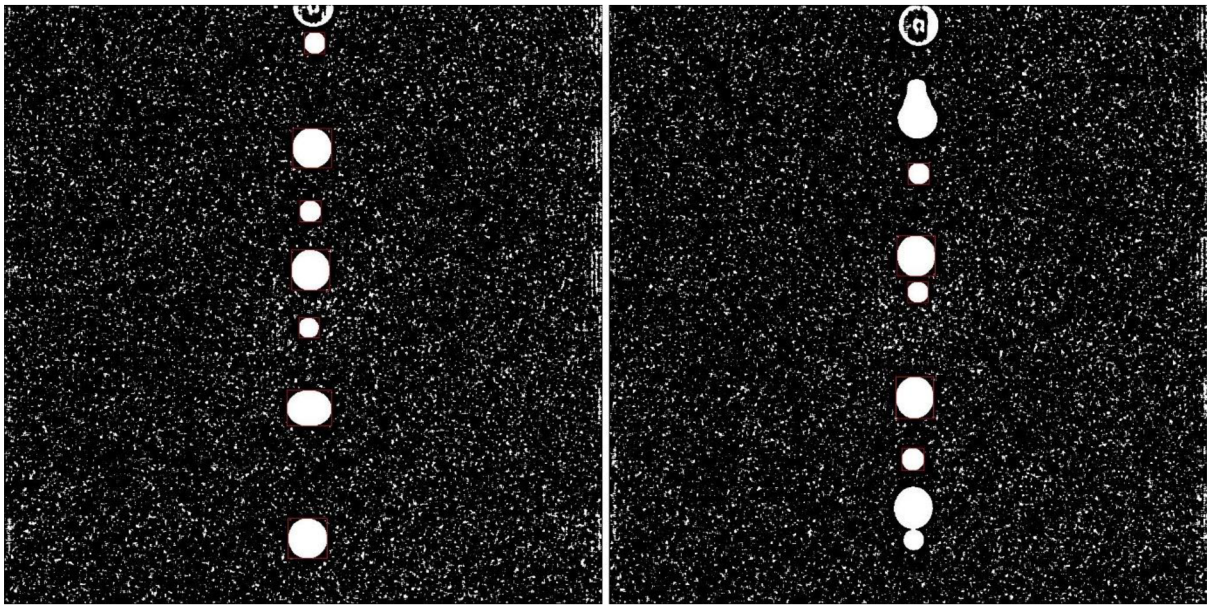
pozn.: podminkaEccMale zde není zmíněna, protože tak malé kapky jsou odfiltrovány podmínkou Area

Při tomto nastavení jsem změřil průměrnou hodnotu průměru kapky 322,2 µm, což je o 4,7% menší hodnota, než 338 µm, získaných pomocí fázové dopplerovské anemometrie. Z grafu relativních četností (obrázek 27) můžeme opět vidět, že má metoda detekovala kapičky o trochu menší.

V histogramech ale krom toho nejsou žádné významné rozdíly.



obrázek 27: histogram dvouvelikostního rozložení



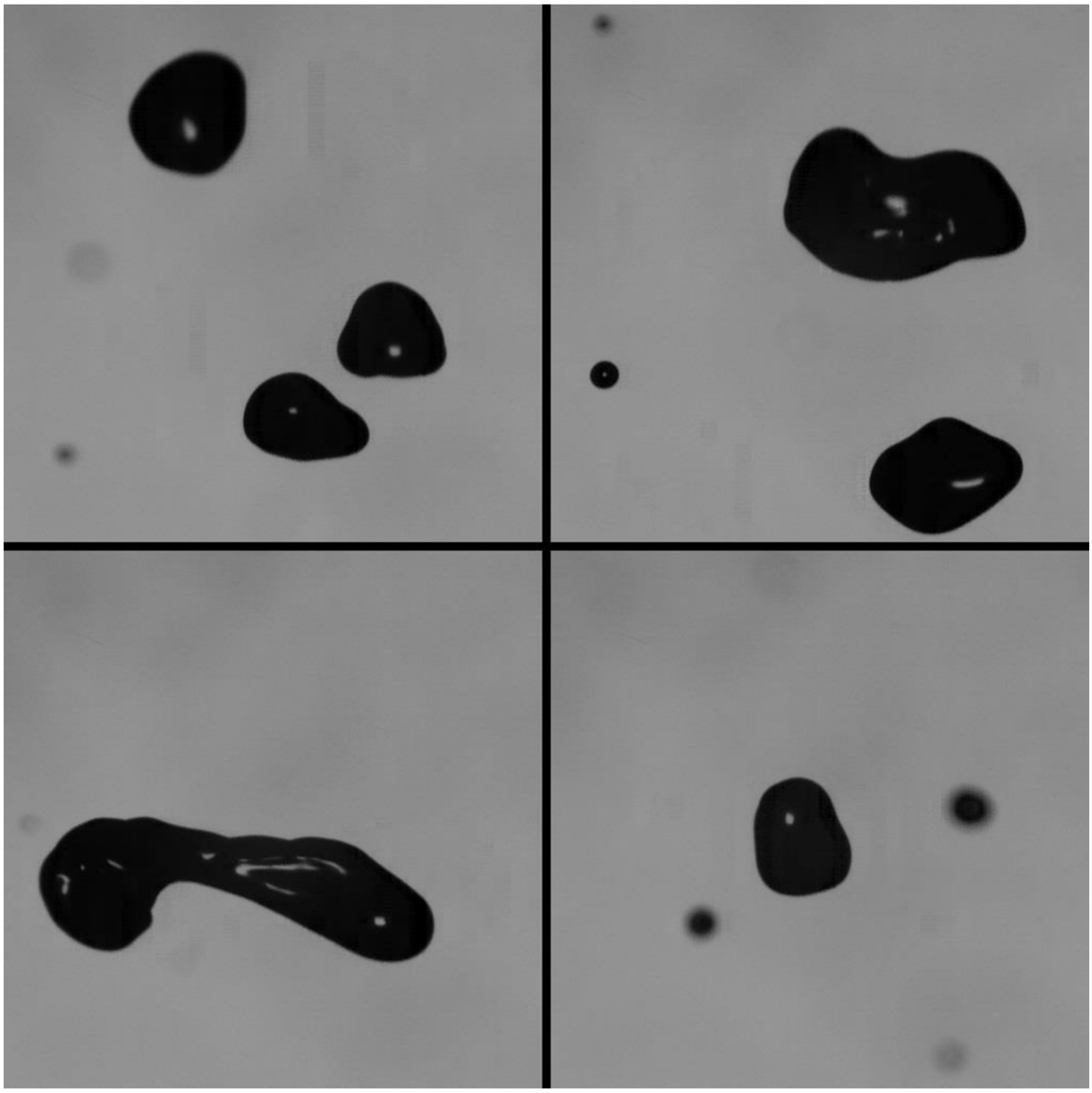
obrázek 28: dva ze zpracovaných snímků, kapičky, které prošly podmínkami jsou v červených rámečcích, na levém obrázku je jedna z excentričtějších kapek, na pravém jsou některé spojené kapky

7.2 Reálný sprej

Svou metodou změřím reálný sprej. Na záznamu (obrázek 29) je šum jen místy, není příliš silný. Kapky jsou různorodé, převážně velké a v různé míře nesférické. Údaje z fázové dopplerovské anemometrie tedy příliš nekorespondují s realitou.

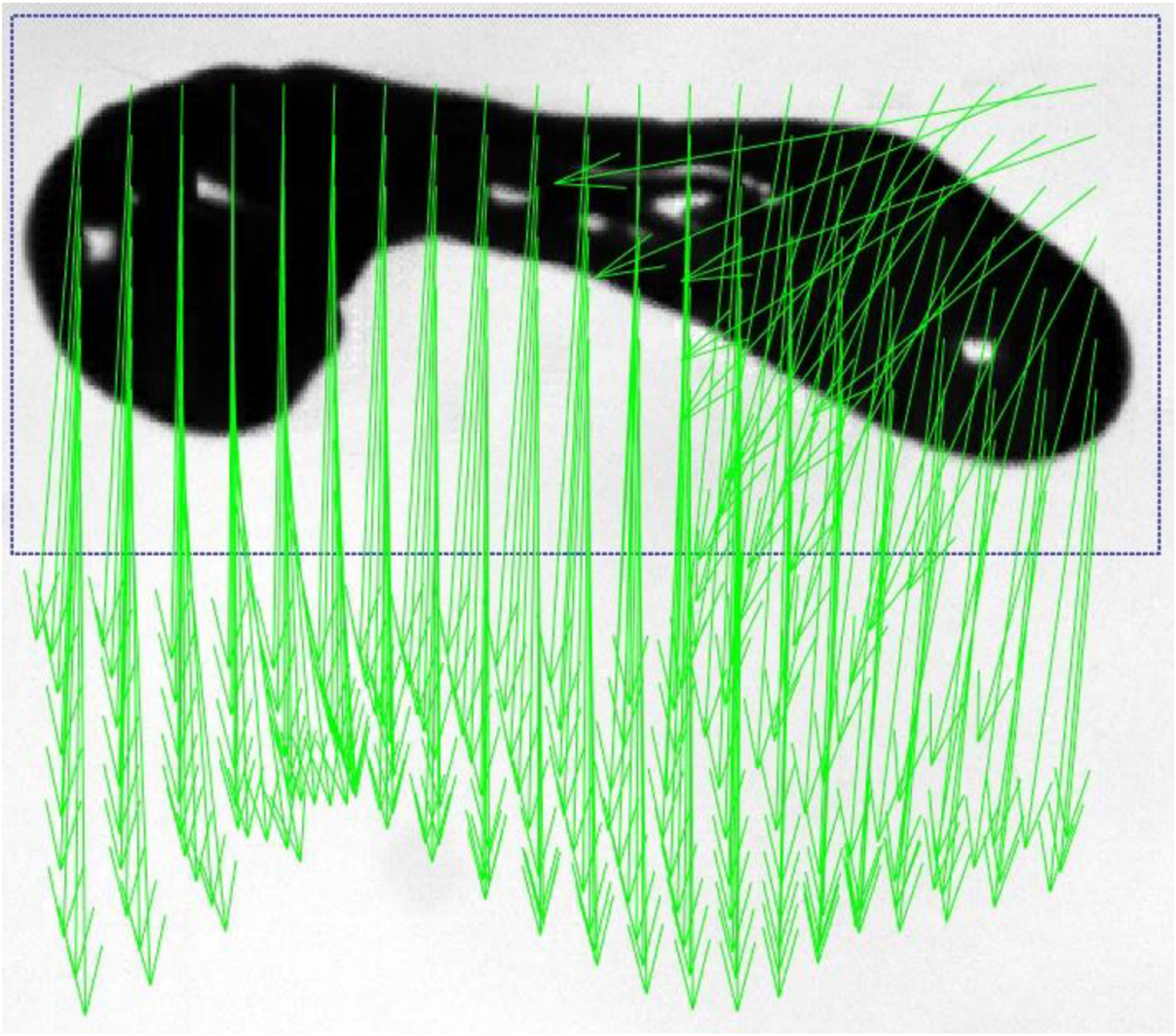
Mým cílem je získat pro tento záznam výsledky. Jejich pravdivost budu kontrolovat vizuálně.

Při pořízení záznamu byla délka impulsu světla 400 ns s 1 000 ns závěrkou, kamera snímala rychlostí 60 000 snímků za sekundu.



obrázek 29: příklady záznamu spreje, který jsem analyzoval

Nejdříve jsem v PIVlabu získal rychlost kapek (viz obrázek 30). S pomocí referenčního obrazu jsem zjistil, že obrázek má asi 73 pixelů na milimetr. Při tomto měřítku se kapičky posunou o 26 pixelů za obrázek, tedy mají rychlost 21,37 metrů za sekundu. Počet obrázků, který by se měl přeskočit, aby bezpečně nedošlo k dvojímu zachycení jedné kapky, je dvacet. Kapky jsou ale relativně velké a pomalé, iterativně jsem určil vhodné číslo přeskočených obrázků jako čtrnáct.



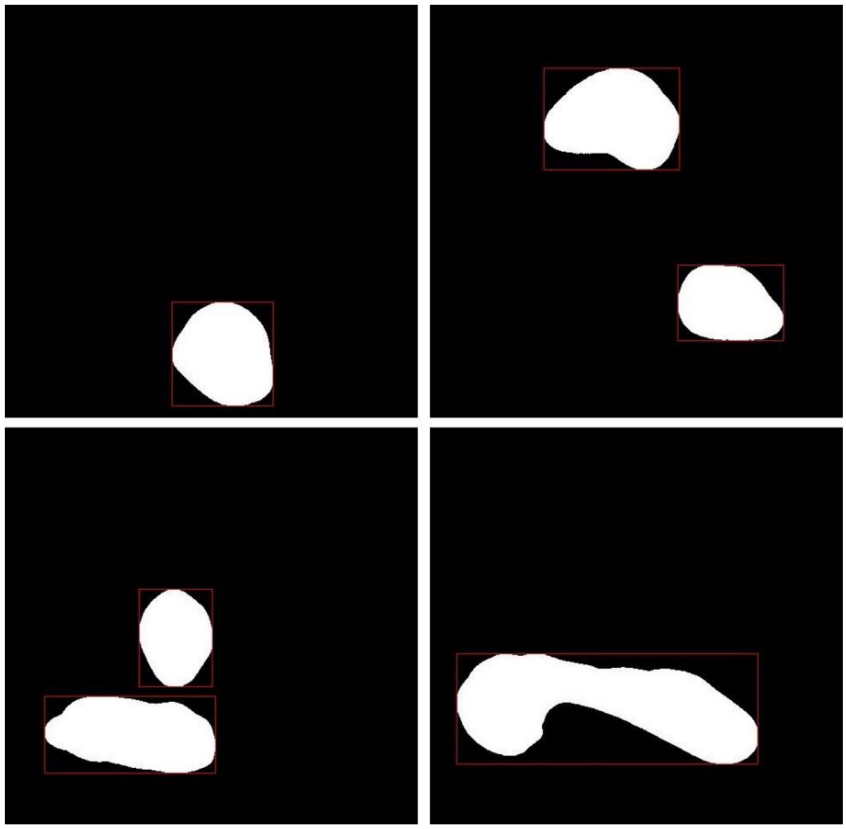
obrázek 30: vektorové pole rychlosti kapky v PIVlabu

Měření bylo provedeno při parametru LoG $\sigma = 0{,}05$, s maskou o velikosti 7×7, mediánový filtr měl masku o velikosti 5×5. Velikosti jednotlivých podmínek byly následující:

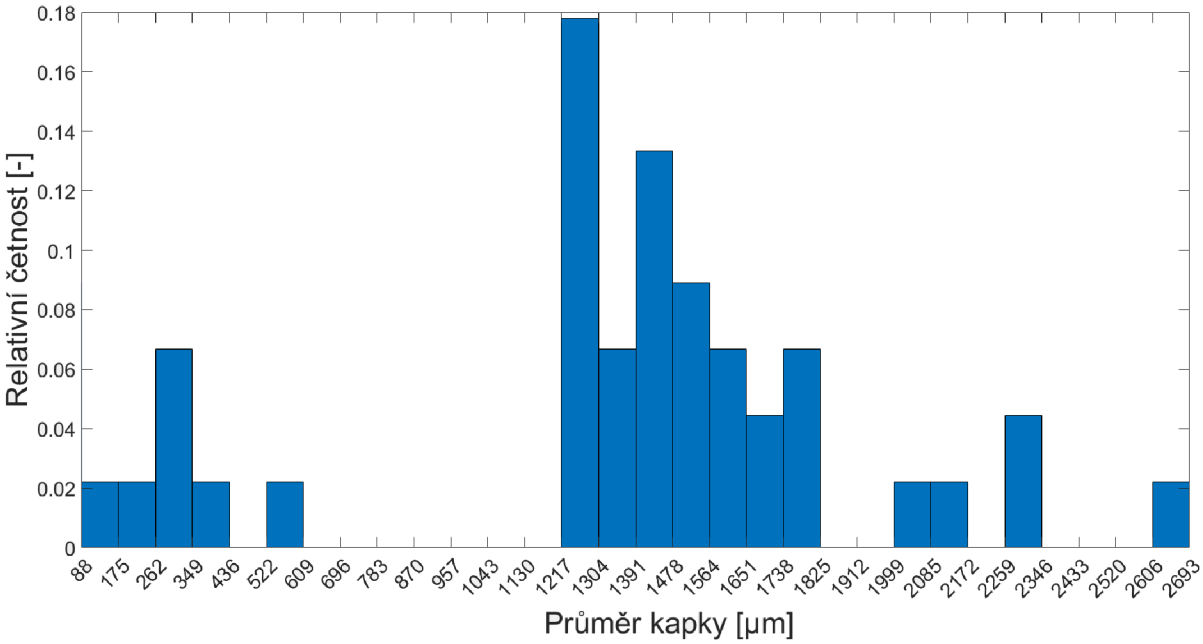
limitEntropie = 0,62
podminkaVyplneni = 0,7
podminkaArea = 20
podminkaEccMale = 0,851
podmínkaEccVelke = 0,99
podminkaKapkaArea = 50
podminkaBBox = 3

Při tomto nastavení jsem změřil průměrnou hodnotu průměru kapky 1481,48 µm. Dohromady jsem změřil 46 kapek, v jednom případě se nejspíše jedná o dvě překryté kapky, jinak zpracování proběhlo bez problémů, viz obrázek 31. Z grafu relativních četností (obrázek 32) můžeme vidět, že většina kapek spreje má rozměry okolo průměru, zhruba ±300 µm. Také zde vidíme nezanedbatelné množství menších kapiček s průměrem do 450 µm. Mezi těmito

dvěma rozmezími se kapky vyskytují velmi vzácně. Ve spreji se relativně často vyskytují také velké kapky s průměrem až do 2700 µm.



obrázek 31: příklady zpracovaného záznamu s reálným sprejem



obrázek 32: histogram reálného spreje

8 Závěr

Cílem této bakalářské práce bylo vytvořit způsob, kterým lze získat rychlost a velikost kapek ze záznamu pomocí počítačového vidění.

Rychlost jsem získal pomocí PIVlabu, který pracuje na bázi metody PIV.

Velikost kapek jsem získal pomocí vlastního kódu v Matlabu.

V rámci rešerše jsem prozkoumal možné přístupy k problematice zpracování obrazu a určil, které z nich jsou pro mou aplikaci vhodné. V praktické části jsem vytvořený postup porovnal s již zavedenou metodou fázové dopplerovské anemometrie a posoudil jeho přesnost, dále jsem změřil rychlost a velikost kapek reálného spreje.

V rámci předzpracování jsem své cíle z velké části splnil. Hranice kapek jsou z obrázku jasně patrné, zároveň mají vyšší intenzitu než ostatní pixely, jsou tedy jednoznačné a dobře segmentovatelné. Tvar kapek odpovídá realitě i u menších kapiček. Jediné, čeho se mi nepodařilo zcela dosáhnout, byla eliminace rozmazaných kapek. Po konzultaci jsem ale usoudil, že tento úkol je bez použití další kamery nesmírně obtížný. I s použitím další kamery není elementární.

Cíle segmentace byly také dosaženy. Každý region je rozlišitelný od ostatních. Kapky byly odděleny od pozadí. Zůstávají ale některé bílé pixely, které nechci detekovat, bylo nutno je odstranit na vyšší úrovni zpracování.

V rámci vyšší úrovně zpracování jsem kvantifikoval záznam. Vytvořil jsem také podmínky, které mají za cíl odfiltrovat zbylé nežádoucí pixely, tyto podmínky byly dostatečné ve všech aplikacích, které jsem v rámci této práce provedl.

Metodu jsem testoval a porovnal s metodou FDA. Obecně se dá říci, že hodnoty získané mojí metodou jsou vždy trochu menší než ty získané pomocí FDA, čím více se ale zmenšují konstanty funkcí předzpracování, tím více se hodnoty blíží. Odchylky v měření průměru nabývají hodnot do 5%. Zdá se také, že má metoda je více citlivá na méně četné druhy kapek.

Nakonec této práce jsem provedl měření na záznamu reálného spreje.

9 Seznam použitých zdrojů

[1] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. *Image processing, analysis, and machine vision*. 4th edition. United States of America: Cengage Learning, 2015, xxxv, 870 stran : ilustrace, grafy ; 23 cm. ISBN 978-1-133-59369-0.

[2] MARQUES, Gonçalo a Rui PITARMA. Industrial Informatics for Quality Assurance and Real-Time Defect Detection Through Computer Vision. SURESH, P., U. SARAVANAKUMAR a Mohammed Saleh HUSSEIN AL SALAMEH, ed., P. SURESH, U. SARAVANAKUMAR, Mohammed HUSSEIN AL SALAMEH. *Advances in Smart System Technologies* [online]. Singapore: Springer Singapore, 2021, s. 325-335 [cit. 2021-05-13]. Advances in Intelligent Systems and Computing. ISBN 978-981-15-5028-7. Dostupné z: doi:10.1007/978-981-15-5029-4_27

[3] SZELISKI, Richard. *Computer Vision* [online]. London: Springer London, 2011 [cit. 2021-05-13]. Texts in Computer Science. ISBN 978-1-84882-934-3. Dostupné z: doi:10.1007/978-1-84882-935-0

[4] HøYE, Toke, Johanna ÄRJE, Kim BJERGE et al. Deep learning and computer vision will transform entomology. *Proceedings of the National Academy of Sciences* [online]. 2021, **118**(2) [cit. 2021-05-13]. ISSN 0027-8424. Dostupné z: doi:10.1073/pnas.2002545117

[5] MARR, David a Ellen HILDRETH. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences* [online]. 1980, **207**(1167), 187-217 [cit. 2021-05-13]. ISSN 0080-4649. Dostupné z: doi:10.1098/rspb.1980.0020

[6] NAGAO, Makoto a Takashi MATSUYAMA. *A Structural Analysis of Complex Aerial Photographs: Advanced applicationsin pattern recognition*. III. Series. New York: Plenum Press, 1980. ISBN 0-306-40571-7.

[7] THIELICKE, William. *The flapping flight of birds: Analysis and application*. [S.l.], 2014. Disertační práce. University of Groningen.

10 Seznam použitých zkratek a symbolů

Δ	laplacián
∇	nabla
σ	charakteristika Gaussova rozdělení
σ^2	rozptyl jasu
FDA	fázová dopplerovská anemometrie
h	maska
LOG	laplacián gaussiánu
PIV	*Particle Image Velocimetry*
x	souřadnice na ose x
y	souřadnice na ose y

11 Seznam příloh

PŘÍLOHA 1 kód matlab