ŽILINSKÁ UNIVERZITA V ŽILINE

AUTOREFERÁT
DIZERTAČNEJ PRÁCE

Žilina, Máj, 2020

Ing., František Kajánek

Žilinská univerzita v Žiline
Fakulta riadenia a informatiky

František Kajánek, Ing.

Autoreferát dizertačnej práce

Počítačové spracovanie obrazu pre vývoj a verifikáciu výpočtových modelov biologických buniek

na získanie akademického titulu **„philosophiae doctor“ (v skratke PhD.)**
v študijnom programe doktorandského štúdia
aplikovaná informatika

v študijnom odbore:
informatika

Žilina, Máj, 2020

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia alebo v externej forme doktorandského štúdia **na Katedre softvérových technológií Fakulte riadenia a informatiky Žilinskej univerzity v Žiline**

Predkladateľ: **Ing. František Kajánek**
Katedra softvérových technológií
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline

Školiteľ: **prof. Mgr. Ivan Cimrák, Dr.**
Katedra softvérových technológií
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline

Oponent: **doc. Ing. Vanda Benešová, PhD.**
Fakulta informatiky a informačných technológií
Slovenská technická univerzita v Bratislave

Oponent: **prof. Ing. Martin Klimo, PhD.**
Katedra informačných sietí
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline

Autoreferát bol rozoslaný dňa: .....

Obhajoba dizertačnej práce sa koná dňa 17.08.2020 o 14.00 h. pred komisiou pre obhajobu dizertačnej práce schválenou pracovnou skupinou odborovej komisie v študijnom odbore **informatika v študijnom programe aplikovaná informatika**, vymenovanou dekanom Fakulty riadenia a informatiky Žilinskej univerzity v Žiline dňa .....

prof. Ing. Karol Matiaško, PhD.
predseda pracovnej skupiny odborovej komisie
študijného programu **aplikovaná informatika**
v študijnom odbore **informatika**

Fakulta riadenia a informatiky
Žilinská univerzita
Univerzitná 8215/1
010 26 Žilina

Úvod

Pri modelovaní toku krvi v mikrofluidických zariadeniach je veľmi dôležité preskúmať správanie rôznych buniek a taktiež jasne zmerať fyzické vlastnosti daného zariadenia. Náročnosť tejto úlohy spôsobuje, že je potrebné veľké množstvo dát na dizajn a testovanie modelu, ktorý korešponduje tomuto zariadeniu. Zbieranie dát na tento účel preto musí byť čo najpresnejšie a čo najefektívnejšie. V dnešnej dobe jednou z možností je automatizované spracovanie na počítačoch.

Dáta z biologických experimentov bývajú najčastejšie dostupné vo forme obrazkov alebo videa. Kvôli množstvu týchto dát je potrebné používať pokročilé metódy počítačového videnia na ich spracovanie. Už existujú rôzne algoritmy na tieto úlohy ale napriek tomu ich využitie alebo dizajn nových je veľmi náročný. V našom prípade potrebujeme komplexný systém schopný analýzy videa, ktorý pomocou detekcie a trasovania buniek získa dáta potrebné na validáciu modelov.

V tejto práci sa zameriavame na problémy ktoré sa vyskytnú pri automatizovanom spracovaní obrazu. Najprv sa zameriame na rýchly prehľad všeobecných metód. Ďalej prejdeme existujúce metódy používané na bunky a vyhodnotíme ich výsledky pri aplikácii na našu úlohy. Potom sa zameriame na detekciu červených krviniek a naše riešenia pre tento problém. Po skončení analýzy a popise riešení prejdeme aktuálne trasovacie algoritmy a naše novo navrhnuté metódy na zlepšenie výsledkov. Na konci v krátkosti zhrnieme budúce možnosti a popíšeme problémy s ktorými sme sa stretli.

1. Zber dát

Modelovanie toku krvi v mikrofluidických zariadeniach je jeden zo spôsobov ako pristúpiť k problémom ktoré sa ťažko riešia inak, ako napríklad vykonávanie biologických experimentov. Výpočtové modely pomáhajú s predikciou výsledkov v prípadoch ktoré je ťažké dosiahnuť v laboratóriu. (Calder, 2018) Taktiež sú užitočným nástrojom pri optimalizácií a dizajne. (Kleineberg, 2017)(Marton, 2017) Na to aby simulácie boli schopné zabezpečiť správne výsledky, je potrebné aby kvalita a detail simulácie bol čo najlepší. V tomto prípade je potrebné správne modelovať elasticitu, interakcie a pohyb červených krviniek, a pretože hematokrit krvi je veľmi vysoký (45%), je veľmi dôležité ich modelovať presne. Modely červených krviniek boli použité na modelovanie procesov v mikrofluidických zariadeniach. (Jančigová, 2014) (Jančigová 2015) Na validáciu modelov zaoberajúcich sa viacerými bunkami je možné použiť experimenty vykonané na jednotlivých bunkách, ktoré následne pomôžu s kalibráciou biomechaniky danej bunky. Ako príklad máme naťahovanie bunky pinzetou v (Dao, 2006).V prípade validácie makroskopických fenoménov ako napríklad bezbunková vrstva je možné použiť experimenty s veľa bunkami, viac informácií je možné nájsť v (Fedosov, 2010).V druhom prípade je kritické spracovať video sekvencie a extrahovať z nich veľké množstvo dát na validáciu ich modelu.

Z dôvodu zvyšovania dostupných dát je nevyhnutné spracovať dáta automatizovanými spôsobmi v rôznych vedeckých odvetviach, nielen za účelom validácie simulácií. Vedci bežne vytvárajú videá z experimentov aby iné tímy boli schopné využiť ich úsilie. Náš záujem leží v bunkových experimentoch a hlavne v experimentoch s červenými krvinkami. Validácia môže byť vykonaná po spracovaní videa. Sú dostupné nástroje na manuálne spracovanie ale tie sú bohužiaľ veľmi neefektívne a časovo náročné a umožňujú spracovať malé percento dostupných dát. Príkladom automatizovaného spracovania je detekcia a trasovanie buniek.

Trasovanie buniek sa dá rozdeliť na dva typy metód. Prvým typom sú metódy párovania kontúr, ktoré vysegmentuje bunky v prvom obraze a následne hľadá ich kontúry v ďalších snímkach. (Dufour, 2011) (Ka, 2013) Toto vyrieši aj segmentačnú úlohu a trasovaciu úlohu zároveň. Druhý typ môže byť charakterizovaný detekciou všetkých buniek

a následné trasovanie nájdených buniek v obrázkoch/videu pomocou časových informácií. V tejto práci sa budeme hlavne zameriavať na úpravy a vylepšenia metód druhého typu.

1.1. Detekcia objektov

Detekcia buniek a hlavne všeobecný problém detekcie objektov je bežnou úlohou ktorá má veľa využití aj mimo trasovania. Napriek tomu, že táto úloha mala veľa zlepšení v posledných rokoch, stále je považovaná za veľmi náročnú kvôli prekrývaniu objektov, čiastočnej neviditeľnosti alebo rotácií objektov. Jedným z riešení tejto úlohy je napríklad Hough-ová transformácia (Kittler, 1987).Táto metóda hľadá jednotlivé geometrické tvary, v našom prípade kruhov a elíps. Napriek horším výsledkom tejto metódy jej hlavnou výhodou je použitie tejto metódy bez potreby strojového učenia ako prvý krok pri automatizovanom spracovaní obrazu.

Väčšina výkonných metód sa sústreďuje na extrakciu vlastností z obrazu, napríklad Haarové vlnky, histogramy alebo HOG deskriptory. Tieto ale nie sú veľmi dobré na všeobecné objekty a vyžadujú veľa testovania pri ich výbere, prípadne dizajne nových. Tieto ale spadajú do kategórie strojového učenia a vyžadujú dataset na ich tréning.

Najnovším prístupom k tejto úlohe sú neurónové siete. Tie počas tréningu automaticky vyberajú extraktor vlastností na každú úlohu. Tie sa vyberajú priamo z dát a umožnili jedny z najlepších výsledkov na úlohách ako klasifikácia obrazu, rozpoznávanie obrazu, detekcia a segmentácia. Neurónové siete podávajú výborné výsledky na úlohách kde je možné získať veľké množstvá trénovacích dát. Detekcia objektov zväčša využíva konvolučné vrstvy na extrakciu vlastností. V posledných rokoch boli konvolučné neurónové siete využité na pokrok v rôznych odvetviach hlavne vďaka zlepšeniu ich schopnosti využívať a propagovať informácie.

Výsledky detekciu môžu byť interpretované pomocou matice zámeny. Jej informácie sa následne dajú využiť na výpočet metriky Precision/Recall. Precision je percento správnych detekcií, a Recall je percento správne nájdených objektov v danom snímku/videu.

1.2. Trasovanie objektov

Trasovanie objektov s nájdenými objektami na vstupe má jednoduchú ideu: spojiť všetky obdĺžniky nájdené počas trasovania do súvislých trás. Trasy je možné následne využiť na rôzne účely, ako napríklad zisťovanie najčastejších trajektórií, získavanie charakteristík objektov, zmeny v tvare a pod.

Jedným zo spôsobov trasovania je získavať informácie o rýchlostiach objektov v danom snímku videa a predvídať ich pozíciu v ďalšom snímku. Príkladom je mechanizmus kľúčovej dierky, kde sa hľadá nasledujúca bunka v okolí predošlej.

Iným prístupom je particle filtering. Je to metóda na odhadovanie vnútorných stavov dynamických systémov v prípade, že máme k dispozícii iba čiastočné informácie o systéme a nastávajú zmeny. V prípade trasovania objektov to môžeme pozorovať ako objekty, ktoré sú veľmi blízko pri sebe s potenciálnym prekrývaním, a vyhodnotenie ktorý objekt je ktorý po tom ako sa rozdelia s najväčšou pravdepodobnosťou.

Vyhodnotenie trasovania nie je triviálnym problémom. Trasovanie objektov pomocou obdĺžnikov objektov je silne závislé na úspešnosti detekcie. Kvôli tomu musí byť detekčný krok súčasťou hocijakej metriky použitej pri kvantifikovaní úspešnosti daného algoritmu. Samotné trasovanie má veľa užitočných metrík. Medzi ne patrí napríklad priemerná dĺžka nájdenej trasy v porovnaní s datasetom. Ďalšou metrikou je percento správne spojených segmentov trás. Jedným z možných prístupov kondenzácie týchto informácií je napríklad Acyclic oriented graphs matching, ktoré funguje na princípe prikladania váh.

1.3. Bežné problémy

Napriek všeobecným riešeniam detekcie a trasovania, bunky majú svoje vlastné problémy. Pri získavaní videí, nie všetky bývajú použiteľné. Video často býva podpriemernej kvality, nasvietenie je veľmi slabé a nie je možné rozoznať farby. Bunky sa zvyknú pohybovať vysokou rýchlosťou a ani dnes nie je jednoduché vytvoriť video s dostatočným počtom snímkov za sekundu, čo spôsobuje rozmazané bunky. Taktiež existuje niekoľko unikátnych udalostí ako delenie buniek a smrť bunky. Vo všeobecnosti kvôli bežnému vysokému počtu buniek, prekrývanie a veľké zhluky sú taktiež veľkým problémom.

1.4. Deskriptory a klasifikátory

Deskriptory a klasifikátory bývajú bežnými zložkami pri spracovaní obrazu. Umožňujú rôzne úlohy ako rozpoznávanie obrazu, detekcia objektov, automatizované značkovanie a filtering. Deskriptor je reprezentácia vlastností dát, v našom prípade vizuálny popis vlastností v obraze. Tieto vlastnosti musia byť veľmi špecifické aby zachytili dôležité informácie v obraze, ako napríklad tvar, farba, textúra alebo pohyb.

Klasifikátory na druhej strane sú algoritmy, ktoré sú schopné klasifikovať dáta. Výsledkom je diskrétna trieda pre dané dáta. Väčšina klasifikátorov je binárnych ale existujú taktiež viactriedové klasifikátory v menšom množstve. Klasifikátory majú na vstupe vektor vlastností, čo v prípade počítačového videnia môže byť snímka videa alebo výstup deskriptora. Jednoduchým príkladom klasifikátora je rozhodovací strom.

1.5. Strojové učenie

Strojové učenie sa zaoberá skúmaním a výtvorom algoritmov, ktoré dokážu spracovať dáta a podávať predikcie. Tieto algoritmy majú robiť rozhodnutia podľa dát, čo je opakom statického programovania, ktoré dokáže odpovedať len jedným spôsobom. Úlohy na ktoré sa používa strojové učenie bývajú príliš jednoduché pre konvenčné programovacie prístupy, hlavne z dôvodov výkonu a času na vývoj. Príklady kde sa používa sú napríklad detekcia malwaru, filtrovanie emailov alebo problémy počítačového videnia.

Existujú dva typy, učenie s dozorom a bez dozoru. V prípade prvého je potrebný dataset pomocou ktorého je vytvorený matematický model. V našom prípade je to dataset obrázkov v ktorých sú orámované červené krvinky. Najčastejšie sa využíva v dvoch typoch problémov, klasifikačných a regresných. Príkladom algoritmov ktoré používajú tento typ učenia sú napríklad Support Vector Machines alebo Lineárna Diskriminačná Analýza, prípadne rozhodovacie stromy. Je to vhodný spôsob pre úlohy kde sa dá získať pravdivá odpoveď.

Učenie bez dozoru je prístup pomocou ktorého sa popisujú neoznačkované dáta. Hlavnou ideou je dávať štruktúru dátam pomocou nejakej funkcie. Z toho ale vyplýva problém netriviálnosti vyhodnotenia presnosti týchto metód. Tieto metódy nevyžadujú žiadny dataset na vstupe. Príkladom tohto prístupu je napríklad k-means, niektoré neurónové siete alebo autoenkódery.

1.6. Problémy strojového učenia

Strojové učenie má veľa rôznych problémov s ktorými sa musí vysporiadať. Medzi tieto problémy napríklad patrí pretrénovanie. Bežnou logickou úvahou je možné si myslieť, že čím viac učenia tým lepšie výsledky. Bohužiaľ ale keďže strojové učenie častokrát pracuje s nadrovinami, logické úvahy v týchto problémoch nie sú ihneď zrejmé. To taktiež súvisí s problémom dimenzií, kde sa nedá vizuálne predstaviť konkrétna nadrovina.

Vizualizácia a reprezentácia modelov a funkcií strojového učenia je taktiež veľkým problémom. Takto vytvorené algoritmy často pracujú s rôznymi váhami priradenými

k rôznym prvkom počas rozhodovania, pomocou ktorých dávajú jednotný výsledok pre danú úlohu. Aj kvôli tomu sa považujú tieto algoritmy za čierne skrinky a riešenie chýb môže byť aj časovo aj výpočtovo náročné. Viac problémov je možné nájsť v (Domingos, 2012).

1.7. Prehľad práce v tejto oblasti

Vizuálne spracovanie buniek je bežné v počítačovom videní. Táto úloha sa používa napríklad na detekciu rôznych chorôb a na pomáhanie profesionálov pri diagnostike pacientov. Bežný prístup je detekcia a viactriedová klasifikácia buniek z ktorej sú možné rôzne interpretácie (napríklad rakovina).

Problém s detekciou rôznych buniek je, že s nárastom tried narastá komplexnosť úlohy a znižuje sa jej úspešnosť. V našom prípade sa chceme sústrediť len na jeden typ bunky, červené krvinky.

V prípade tradičný metód, morfologické metódy sú často úspešné. Napríklad detekcia hrán pomocou canny edge detektora býva najúspešnejší. Tieto metódy sa následne spoja s už spomínanou Hough transformáciou, ktorá úspešne nájde bunky. Výstup sa môže taktiež použiť na klasifikáciu medzi bežnými bunkami a kosákovitými bunkami. Táto metóda má ale problémy so zlým nasvieteným, ktoré je bežné napríklad v našom datasete.

Ďalším prístupom je použitie histogramov a thresholdingu na detekciu. Rôzne farebné intervaly môžu byť thresholdované na 255, ktoré sa budú považovať za bunky a niektoré na 0, ktoré budú pozadím. Daný obraz je následne spracovaný a hľadajú sa zhluky 255 hodnôt v pixeloch. Zhluky pod 100 pixelov sa považujú za šum. Tieto zhluky sa následne spočítajú a umožňujú relatívne presné počítanie buniek.

V poslednej dobe sa pozornosť upriamila hlavne na využívanie neurónových sietí. Napríklad v (Reyes-Aldasoro, 2007) sa používajú aj na detekciu aj trasovanie. Na trasovanie sa používajú fyzické vlastnosti bunky ktoré následne umožňujú spájať detekcie do trás. Na druhej strane v (Heikkila, 2017) sa používajú dve úplne rôzne neurónové siete. Prvá slúži na detekciu, a druhá sa pomocou online tréningu naučí hľadať jednu špecifickú bunku.

V (Yi, 2017) je popísaný spôsob kde pomocou manuálnej anotácie a využitia samplovacích metód natrénuje bunka na jednu špecifickú bunku. Následne sa použite particle filtering metóda počas trasovania, ktorej výstupy sa použijú ako vstup pre inú neurónovú sieť, ktorá dáva pravdepodobnosť nálezu tej istej bunky. Oproti predošlému príkladu používa toto riešenie úplne iný dizajn neurónovej siete.

Pokiaľ sa pozrieme na state-of-the-art algoritmy v tejto oblasti, bývajú to väčšinou prístupy s istou úrovňou redundancie, kde sa používa nielen neurónová sieť ale taktiež nejaká tradičná metóda, ako už napríklad spomínané morfologické metódy. V trasovacom kroku je často používaný jednoduchý prístup bez strojového učenia a táto oblasť ešte nie je veľmi preskúmaná.

Mnoho metód v tejto oblasti taktiež nepracuje špecificky s červenými krvinkami a taktiež často pracujú s ideálnymi dátami. Vo všeobecnosti ale metódy, ktoré sa sústreďujú na jeden typ buniek, dávajú lepšie výsledky, ale iba v detekčnom kroku. Dôvodom nedostatku výskumu o hľadaní červených krviniek môže byť napríklad nedostatok datasetov, ktoré obsahujú výhradne červené krvinky. Napriek tomu je sľubné preskúmať metódy, ktoré boli použité na iné typy úloh a taktiež vyskúšať deep learning na trasovacích úlohách.

2. Doterajší výskum

Zlepšenia v tejto oblasti boli urobené taktiež v našom tíme. V (Mučka, 2017) (Tomášiková, 2017) bol vytvorený algoritmus, ktorý sa skladá z dvoch častí: detekcie a trasovania.

Prvý krok tohto algoritmu deteguje červené krvinky. Na vstupe sú statické obrázky alebo video. Výstupom tohto kroku sú stredy buniek v danom snímku/obrázku. Je tu využitá

metóda odstránenia pozadia, pomocou ktorého sa odstráni šum a nepotrebné prvky. Následne sa tento snímok spracuje pomocou detektora hrán, v tomto prípade canny edge detector. Výsledné hrany sa použijú pomocou Hough transformácie na hľadanie kruhov a elíps. Hough transformácia nájde zhluky bodov vysokej intenzity v strede tvarov a následne pomocou thresholdingu sa klasifikujú dané stredy ako stredy buniek.

Druhý krok má na vstupe výstup prvého kroku. Jednotlivé bunky sú počas neho spojené do trás pre každú bunku. V prvotnom kroku, kde nie sú dostupné žiadne fyzické dáta sa hľadajú okolité nálezy stredov. V následných krokoch algoritmus používa informácie z predošlých snímkov na predikciu smeru pohybu v budúcnosti. Na tento účel sa používa toková matica kvapaliny v každom pixeli. Toto je dôležité v prípade vysokej fragmentácie trasy a taktiež v prípade záhybov a prekážok. Výstupom tohto kroku sú trasy, ktoré môžu byť následne použité na rôzne účely. Aktuálna implementácia dokáže preskakovať snímky s chýbajúcou detekciou.

Hlavným problémom tohto prístupu je, že každý krok má veľa problémov. Detekcia zlyháva pri zlom nasvietení a nejasných a neúplných tvaroch. Trasovanie následne aj pri perfektnej detekcií občas zlyhá kvôli prelínaniu buniek. Na vývoj a porovnávanie bolo nutné výhradne využívať anotované dáta. V tejto práci budeme následne pojednávať o vylepšeniach detekčného a trasovacieho kroku a taktiež o extrakcí dát z každého kroku za účelom validácie.

2.1 Analýza aktuálnej situácie

Tak ako bolo spomenuté v prehľade, existuje veľa rôznych algoritmov na detekciu buniek a je jasný posun v tejto oblasti. Problém je časté používanie ideálnych dát s perfektným nasvietením. V našom prípade začali problémy pri spracovaní iných videí. Thresholding a detekcia hrán z rôznych dôvodov zlyhávala a detekcia mala ešte horšie výsledky. Taktiež tam nastávali nové unikátne problémy ako zdvojené hrany ktoré spôsobovali, že Hough transformácia nesprávne hľadala stredy. Taktiež v prípade veľmi podobných kruhových prekážok ich klasifikovala ako bunky.

Ak sa pozrieme spätne na predošlú analýzu, riešením týchto problémov bolo buď a) použiť viaceré metódy ktoré sa navzájom dopĺňali, alebo b) použiť strojové učenie. My sme zvolili prístup strojového učenia. Strojové učenie v dnešnej dobe ponúka veľa nových prístupov a jedny z najlepších detekčných výsledkov. Keďže sme zostavili dataset za účelom vyhodnocovania presnosti našich algoritmov, táto hlavná prekážka pre nás odpadla. Tento dataset mal ale trochu odlišný formát ako výstupný formát, keďže anotácia buniek produkuje obdĺžniky a nie len stredy buniek.

Ďalším dôvodom na výber strojového učenia bola predošlá skúsenosť s používaním a implementáciou nových častí pre klasifikačný meta algoritmus AdaBoost s použitím SHOG deskriptorov a Viola-Jones kaskády, a taktiež skúsenosť s architektúrou CUDA, GPU architektúrami a Support Vector Machines.

2.2 AdaBoost

Naše predošlé skúsenosti počas diplomovej práce nás viedlo k AdaBoost algoritmu a vyhodnotiť jeho výsledky na našej úlohe. AdaBoost je meta-algoritmus strojového učenia, ktorý môže byť použitý na rôzne úlohy. Meta časť sa týka trénovania nejakého jednoduchšieho slabého klasifikátora, napríklad rozhodovacích stromov. AdaBoost ich používa veľké množstvo a následne ich spája do jedného silného klasifikátora.

Vstupom tohto algoritmu je vektor vlastností a na výstupe je klasifikácia na určitú triedu. Počas tréningu sú požadované označené dáta podľa triedy, na ktorých sa natrénuje matematický model. V testovacej fáze následne akceptuje vektory vzoriek o ktorých potrebujeme rozhodnúť.

Je možné používať rôzne deskriptory. Prvotný AdaBoost bol použitý dokopy s Haarovými vlnkami a kaskádou na detekciu tvárí. Kaskáda je použitie viacerých klasifikátorov v tandeme za účelom zvýšenia priepustnosti algoritmu. Na začiatku sa zvyknú použiť rýchle klasifikátory a na konci pomalé ktoré značne zlepšia výsledok. V našom prípade sa používa Viola-Jones kaskáda na tento účel. Táto kaskáda podporuje 2 rôzne deskriptory, Haarové vlnky a Local Binary Patterns. V našom prípade sme chceli vyskúšať Haarové vlnky.

Kaskádu sme počas tréningu nakonfigurovali na 25x25px obrázky, ktoré sme vysegmentovali z nášho datasetu. Zvolili sme jednoduché 0 hĺbkové rozhodovacie stromy bez limitu na počet slabých klasifikátorov.

V najlepšom prípade po vykonaní rôznych testov sa nám podarilo dosiahnuť 50% nájdených všetkých krviniek v snímkach datasetu. Taktiež sme mali 50% chybovosť nálezov, tzn. polovica klasifikovaných objektov bolo nesprávnych, čo bolo spôsobené zlou lokalizáciou nálezov a taktiež nálezmi v nelogických oblastiach. Z toho dôvodu jediné využitie tohto algoritmu by bolo v spojení s Hough transformáciou.

Napriek tomu, že tieto výsledky sú relatívne úspešné, rozhodli sme sa pokračovať s inými prístupmi strojového učenia, špecificky deep learningu.

3. Dataset

Za účelom jednoduchého vyhodnocovania algoritmov, či už detekcie alebo trasovania, sme zostavili dataset. V aktuálnej verzií máme v datasete snímky dvoch videí. V prvom videu máme anotovaných 300 snímkov na detekciu a trasovanie. V tomto videu je v priemere 50 buniek na snímku a bunky majú medzi 25-35px a rozlíšenie videa je 1280x720px. V druhom videu máme anotovaných 100 snímkov na detekciu. V tomto videu je v priemere 35 buniek na snímku, majú medzi 45-55px a rozlíšenie videa je 512x512px.

Hlavným cieľom nášho datasetu je mať rozmanitú vzorku červených krviniek a aby boli zachytené všetky možné situácie v dostatočnom počte. Napríklad druhé video má nielen úplne rôzne charakteristiky buniek, ale má taktiež prekážky v ktorých sa bunky deformujú. Na druhej strane sú v tomto videu bunky omnoho jasnejšie viditeľné. Do budúcna je snaha tento dataset naďalej rozširovať. Samotné dáta sú uložené vo formáte XML ktorý obsahuje informácie o danej bunke a taktiež ku ktorému snímku patria.

4. Konvolučné neurónové siete

V prípade testovania konvolučných neurónových sietí bolo ihneď jasné, že nechceme vytvárať vlastnú implementáciu. Z tohto dôvodu sme sa rozhodli vyhľadať voľne dostupnú implementáciu ktorá spĺňa naše potreby na experimentovanie s novými dizajnami. Naše doterajšie algoritmy boli v Pythone a C++, čiže to zúžilo náš výber. Z toho dôvodu sme zvažovali TensorFlow a PyTorch.

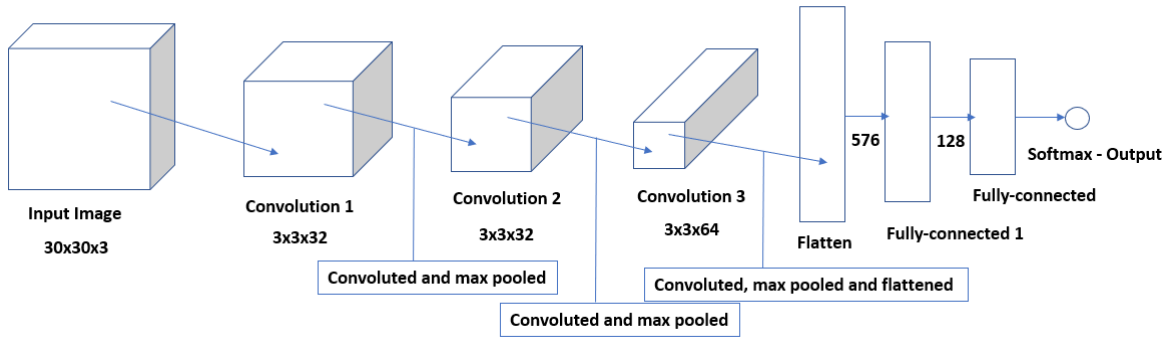
Oba tieto frameworky podporujú experimentovanie v Pythone a implementovanie v C++. TensorFlow bol v roku 2018 viac rozšírený a bolo naň dostupných viac existujúcich návodov, zatiaľ čo v tom čase PyTorch bol stále v beta verzií. Z toho dôvodu sme sa rozhodli pracovať v TensorFlowe s možnosťou vyskúšať PyTorch v budúcnosti. Oba tieto frameworky sú open-source a umožňujú relatívne jednoduché rozširovanie.

4.1 Prototyp

Ako prvotný prototyp práce s konvolúčnymi neurónovými sieťami sme vytvorili jednoduchý dizajn pomocou Pythonu v TensorFlowe. Na vstupe tejto siete je snímka z videa a na výstupe pravdepodobnosť, že daná vzorka obsahuje červenú krvinku. Tento prototyp sa skladá z troch konvolučných vrstiev, jednej vyrovnávacej vrstvy a dvoch plne spojených

vrstiev. Následne sme použili Adam Optimizer, ktorý minimalizuje chybu na anotovaných dátach. Tento algoritmus sme vyhodnocovali na 1500 iteráciách, hlavne z dôvodu fungovania čiste na CPU v tejto verzii.

Konvolúčne vrstvy aplikujú konvolúčny operáciu. Každá vrstva aplikuje 3x3 operáciu s max poolingom. Vďaka tomu zredukuje výstupný priestor a taktiež zvýši rozhľad siete. Prvá a druhá vrstva produkuje 32 kanálov a tretia 64 kanálov. Posuny vrstiev sú 1,2,2,1 v 4D tensore. Po týchto vrstvách je vyrovnávacia vrstva ktorá pretransformuje 4D tensor na 1D tensor. Ten je následne vstupov pre prvú plne spojenú vrstvu, ktorá má 128 hodnôt na výstupe, a následne finálna vrstva, ktorá má 2 hodnoty na výstupe, pravdepodobnosti, že daná vzorka je bunka alebo nie je bunka.



Adam optimizer je stochastická gradientová metóda, ktorá udržiava jednotnú mieru učenia na všetky úpravy váh a má adaptívnu mieru učenia na váhy siete. Je to bežne používaný algoritmus v deep learningu.

Trénovací dataset sa skladá z 279 obrázkov buniek a 549 obrázkov pozadia, ktoré boli vysegmentované z nášho datasetu. Testovací dataset sa skladá z 261 obrázkov buniek a 873 obrázkov pozadia. Všetky tieto vzorky boli náhodne vybrané z nášho datasetu a tieto podmnožiny sa navzájom neprelínajú. Výsledná úspešnosť tejto siete je okolo 87% nájdených buniek na testovacích dátach. Napriek relatívne krátkemu tréningu, oproti našim doterajším výsledkom detekcie sú to veľmi sľubné výsledky.

Na vyhodnotenie tohto prototypu sme použili všetky zvyšné dáta v datasete. V tomto prípade to bolo 7230 vzoriek buniek a 116127 pozadí. Finálne výsledky boli 79% precision a 85% recall. Ak porovnáme výsledky s AdaBoostom, ktorý mal 50% precision a 50% recall, a Hough Transformáciou ktorá mala 90% precision a 65% recall, tak ihneď dostávame veľmi kvalitné výsledky.

Náš prototyp sa dá zlepšiť pomocou nových poznatkov zo state-of-the-art algoritmov. Namiesto používania poolingových vrstiev a konvenčných konvolúčnych vrstiev, môžeme využiť husté vrstvy ako popísané v (Weinberger, 2016). Tie umožňujú zlepšiť výkonnosť propagovaním informácií tak aby sa nestratili. Okrem toho môžeme odstrániť jednu plne spojenú vrstvu podľa poznatkov z aktuálnej literatúry.

Finálna verzia sa skladá zo siedmych hustých konvolúčnych vrstiev, jednej poolingovej vrstvy a jednej plne spojenej vrstvy. Finálna verzia bola trénovaná s 10000 iteráciami a identický tréning bol urobený aj s prvotným prototypom. Vďaka pridanému tréningu a lepšiemu spracovaniu dát na vstupe dosahoval prototyp 97% precision a 93% recall, a finálna verzia o niečo lepších 98% precision a 93% recall.

Tieto prototypy majú ale jednu veľkú chybu. Nie je možné s nimi spracovávať simultánne celé snímky videa a spracovanie trvá veľa výpočtového času. Z toho dôvodu sa musíme pozrieť na prístupy, ktoré umožňujú efektívne spracovanie celých snímkov videa.

4.2 Testovacia konfigurácia

Na vyhodnotenie detekcie sme museli predspracovať náš dataset. Rozdelili sme ho do nasledovných podmnožín:

1. 150 snímkov prvého videa na trénovanie a 50 snímkov prvého videa na testovanie
2. 200 snímkov prvého videa na trénovanie a 50 snímkov prvého videa na testovanie
3. 200 snímkov prvého videa na trénovanie, 50 snímkov prvého videa na testovanie, 30 snímkov druhého videa na trénovanie a 20 snímkov na testovanie (táto podmnožina bola taktiež 3x prehádzaná so zvyškom dát za účelom cross validácie)

Dôvodom na rôzne podmnožiny je zistiť dopad dát na presnosť neurónovej siete. Špecifické skupiny dataset sa neprelínajú. Trénovací proces má na vstupe celé snímky s anotovanými červenými krvinkami. Zvyšok obrázku sa používa na vzorkovanie pozadí.

Neurónové siete boli trénované so 100000 iteráciami, ale taktiež sme vyskúšali konfiguráciu s 200000 aby sme zistili či je to dostatočné trvanie tréningu. Všetky konfigurácie sietí boli zvolené na COCO dataset pred našimi úpravami, aby sme mali spoločný menovateľ pre všetky testy. Tento dataset je veľmi rozmanitý a niektoré prvky sú podobné bunkám. Našimi hlavnými zmenami bolo upraviť resizer na veľkosti našich videí. Boli testované RGB a grayscale obrázky počas tréningu a testovania za účelom zistenia dopadu farby na výsledky. V testovacej fáze je vstupom jedna snímka/obrázok videa. Keďže sme testovali grayscale obrázky, počas testovania bolo nutné duplikovať sivý kanál na 3 z architekturálnych dôvodov.

Náš testovací počítač sa skladal z Threadrippera 2950X a GTX 1080 Ti grafickej karty. Trénovací čas väčšiny modelov bolo okolo 12 hodín, v najhorších prípadoch okolo 24h na jeden model.

4.3 Architektúry

Na testovanie sme zvolili 3 state-of-the-art architektúry:

1. SSD – Single Shot Detector
2. Faster R-CNN – Region – Convolutional Neural Network
3. R-FCN – Region-based Fully Convolutional Network

Hlavným výberovým kritériom boli dostupnosť týchto architektúr a taktiež ich vysoká presnosť na datasetoch ako MNIST, CIFAR alebo SVHN.

SSD architektúra využíva kotvový systém pomocou ktorého následne klasifikuje obdĺžnik tej danej kotvy. Je hlavne smerovaná na rýchlosť a má zlý výkon pri hľadaní malých objektov, ale tento limit sa dá upraviť pomocou nastavenia parametrov. Na druhej strane Faster R-CNN a R-FCN sú siete, ktoré odhadujú zaujímavé regióny a následne klasifikujú len tieto regióny. SSD namiesto toho pracuje s omnoho zredukovanými obrázkami. Napriek tomu, že druhé dve siete sú si podobné v hlavnej idei, ich správanie na rôznych úlohách spôsobuje, že musíme vyhodnotiť na našej úlohe obe.

4.4 Výsledky testov

Na vyhodnotenie sietí sme použili metriky precision a recall. Pri rozhodovaní či daný obdĺžnik z neurónovej siete sa zhoduje s obdĺžnikom z anotovaných dát sme používali Intersection over Union (IoU) princíp prelínania s koeficientom 0.5. Okrem toho sme mali druhý parameter ktorý špecifikoval maximálnu vzdialenosť stredov obdĺžnikov a tam sme nastavili hranicu na 0.5 * šírka anotovaného obdĺžnika. Pri vyhodnocovaní týchto kritérií sme sledovali IoU hodnoty medzi 0.5, 0.3 a 0.1 zatiaľ čo pri vzdialenostiach sme sledovali hodnoty medzi 0.5 a 0.3. Tie nám dávali ďalšiu rozhodovaciu silu pri posudzovaní lokalizácie nálezov.

Najprv sme otestovali všetky 3 architektúry na všetkých 3ch podmnožinách datasetu. Výsledky je možné vidieť v tabuľke. Tu môžme vidieť, že plne natrénovaná sieť podáva omnoho lepšie výsledky ako naše existujúce metódy. SSD sa správa zle na našom datasete, napriek rovnakej konfigurácií ako zvyšné dve metódy.

	200 snímkov		250 snímkov		250 a 50 snímkov	
Precision/Recall	Prvé video	Druhé video	Prvé video	Druhé video	Prvé video	Druhé video
SSD	90.9%/66.3%	16.7%/11.0%	94.5%/68.0%	0.0%/0.0%	95.1%/68.7%	99.2%/93.3%
SSD – upravené	-	-	-	-	98.7%/88.0%	99.2%/95.1%
Faster R-CNN	99.6%/89.7%	0.0%/0.0%	99.3%/97.2%	1.4%/0.1%	99.5%/94.7%	98.7%/98.7%
R-FCN	99.2%/82.1%	2.7%/0.4%	99.6%/88.6%	10.0%/0.4%	82.5%/86.5%	95.5%/97.4%

Taktiež si môžeme všimnúť, že siete podávajú omnoho horšie výsledky pri použití 200 snímkov. To znamená že budúce rozširovanie trénovacích dát je kritické na lepšie výsledky. Taktiež si môžeme všimnúť, že aj po pridaní druhého videa do tréningu, napriek jemnému zníženiu recallu, precision sa znova zlepšil. Výkon na druhom videu aj po pridaní iba 50 snímkov je omnoho lepší ako na prvom videu, čo je pravdepodobne kvôli väčším bunkám, jasnejšom okraji buniek a neexistujúcom prelínaní. SSD na druhom videu podáva omnoho lepšie výsledky, čo iba zosilňuje teóriu o probléme s malými objektami. R-FCN na druhej strane sa správa horšie po pridaní druhého videa viac ako SSD a Faster R-CNN.

Kým prejdeme naše navrhované úpravy SSD za cieľom lepších výsledkov, musíme vysvetliť dopad našich metrík. Náš recall je vysoký, pretože sme použili vzdialenostný koeficient 0.5 v správnych výsledkoch. Napriek tomu že tieto výsledky sú horšie lokalizované, sú to stále validné detekcie. Toto je zjavné hlavne v prípade SSD ktoré má väčší problém s hľadaním špecifického miesta danej bunky. Pri znížení koeficientu na 0.3 sa zhorší recall SSD o 16% na prvom videu. Po úpravách SSD modelu, kde sme upravili parametre kotiev na menšie objekty a zvýšili pracovné rozlíšenie z 300px na 600px sa nám podarilo opraviť nielen lokalizačné problémy na prvom videu ale taktiež aj celkovú degradáciu výkonu oproti zvyšným architektúram. Jediným dopadom bolo, že tréning tejto siete sa zdvojnásobil, čo ale pre nás nie je problém.

Ďalej sme vyhodnotili 200000 iterácií pri tréningu. Vo všetkých troch prípadoch siete trpeli z pretrénovania a degradácie precision. To znamená že nami zvolený počet iterácií bol správny. Následne sme sa pozreli na vyhodnotenie odstránenia pozadia. Vo veľa metódach toto umožní odseparovať šum a zlepšiť presnosť modelu bez ľudského zásahu. V prípade SSD architektúry to malo za výsledok detekciu v druhom videu bez použitia druhého videa na tréning. Na druhej strane odstránenie pozadia taktiež spôsobilo degradáciu výsledkov na prvom videu, takže nemôžme vyhodnotiť túto úpravu ako jednoznačnú. Situáciu taktiež komplikuje neúmyselný posun kamery v prvom videu, ktorý znemožňuje používanie statického pozadia pre celé video.

Ďalším experimentom bolo vyhodnotenie využitia grayscale obrázkov na tréning. Tu si môžeme všimnúť, že R-FCN má silne degradované výsledky po odstránení farby počas tréningu a je kvôli tomu nepoužiteľný s touto úpravou. Ďalej si môžeme všimnúť že Faster R-CNN a SSD majú na prvom videu síce horší precision ale na druhej strane lepší recall. Na druhom videu dokonca SSD má lepšie obe metriky. Ak následne použijeme grayscale model na farebných obrázkoch, dostaneme skoro identické výsledky. To nás vedie k záveru, že farba je dôležitá pri detekcií napriek tomu, že na prvý pohľad snímky vyzerajú prevažne sivé.

	Prvé video - gray	Druhé video - gray	Prvé video - RGB	Druhé video - RGB
SSD	97.2%/88.3%	99.4%/95.5%	97.7%/87.6%	99.4%/95.5%
Faster R-CNN	98.2%/95.9%	99.4%/96.6%	98.2%/95.2%	99.4%/96.6%
R-FCN	72.7%/80.9%	87.9%/96.9%	71.8%/73.1%	87.9%/96.9%

Po zhodnotení výsledkov sme si všimli, že v niektorých prípadoch neurónové siete prekonávajú dokonca ľudí. Počas diagnostikovania chýb sme našli niekoľko chýbajúcich anotácií v našom datasete, čo následne zlepšilo výsledky sietí. Taktiež sme si všimli že niektoré nenájdené bunky boli nájdené, ale povedzme pre zhluk troch našla sieť jeden zle lokalizovaný v strede. Toto je omnoho lepšie ako keby sieť podávala náhodné výstupy.

Ako poslednú kontrolu sme vykonali 4-násobnú cross-validáciu na poslednej podmnožine. V prípade Faster R-CNN sme dosahovali 99% ± 1% precision a 94% ± 4% recall. V prípade opraveného SSD sme dosahovali 97% ± 1% precision a 80% ± 4% recall. Nakoniec v prípade R-FCN sme dosahovali 99% ± 1% precision a 86% ± 2% recall. Tu si môžeme všimnúť, že SSD má viac nestabilný výkon ako R-FCN.

5. Trasovanie a bunky

V predošlých kapitolách sme prešli algoritmus používaný na trasovanie buniek. Taktiež sme spomenuli parameter po ktorý umožňujeme preskakovať snímky. Na vytvorenie spoločného základu sme vyhodnotili trasovací dataset s parametrom 5. Výsledky možno vidieť v tabuľke. Použili sme 300 snímkov nášho trasovacieho datasetu na tieto výsledky. Brali sme do úvahy, že anotované dáta majú recall 100%, a v prípade detegovaných dát sa to zníži na 95%. Týchto 5-6% spôsobuje cca 0.6 dier na jednu trasu. Výsledky tohto algoritmu sú už teraz veľmi sľubné.

Priemerná dĺžka trasy	Prechod algoritmu		
Zdroj dát	Prvý	Druhý	Fragmenty na trasu
Anotované	41.602/355	59.552/248	1 : 2.3
Detegované	36.243/399	48.5/298	1 : 2.9

5.1 Toková matica

Prvým problémom existujúceho algoritmu je jeho využitie tokovej matice. Aktuálne sa táto toková matica vytvára z existujúcich trás ktoré už boli detegované. Toto spôsobuje riedkosť a nerovnomernosť dát v tejto tokovej matici. Za účelom vyriešenia tohto problému sme spustili výpočet toku kvapaliny v namodelovanom kanáli tohto tvaru. Na jej vstupe sme museli nadefinovať geometriu kanálu a rýchlosti kvapaliny na začiatku. Prvé sme získali z videa a druhé sme boli schopní nadefinovať pomocou pohybu jednotlivých buniek z anotovaných dát.

S aktuálnymi výsledkami sme boli schopný vyextrahovať tieto dáta. Najrýchlejšia bunka sa pohybovala okolo 11 pixelov za frame. Takéto rýchlosti sa vyskytujú presne v strede kanála prvého videa. Simuláciu sme vyhodnotili pomocou PyOIF modulu softvéru ESPResSo. Bunky boli skonštruované v rozmedziach 25-35px tak ako vo videu.

Ako výsledok simulácie sme boli schopní získať perfektnú tokovú maticu s relatívne malou odchýlkou. S touto tokovou maticou sme taktiež získali pozície buniek a ich rýchlostí na ďalšie experimenty.

Po vyhodnotení existujúceho algoritmu s touto tokovou maticou, dostali sme výsledky s odchýlkou 0.5 od už spomínaných hodnôt. Toto nám dáva najavo, že táto toková matica je dosť presná aby nebola prekážkou pre tento algoritmus.

5.2 Trasovanie neurónovou sieťou

Ďalším našim krokom bola snaha vylepšiť výkon trasovania a kvôli tomu sme nadizajnovali prototyp pomocou konvolučných neurónových sietí. Snaha je zobrať výstup aktuálneho algoritmu a vylepšiť ho pospájaním niektorých trás. Sieť má za účel predikovať pozíciu v ďalšom snímku presnejšie ako by to urobil fyzikálny model.

V našom prvom kroku sme skonvertovali náš dataset na použiteľný vstup pre našu sieť. Vytvorili sme 30x30x5 matice z nášho trasovacieho datasetu, zacentrované na každú bunku, ku ktorej sa snažíme nájsť pozíciu v budúcnosti. Prvé 2 kanály obsahujú zložky rýchlosti vektora X a Y danej bunky na súradniciach [15,15] a zvyšok samé 0. V 3ťom kanáli sa nachádzajú všetky stredy buniek v 30x30px okolí bunky. Posledné dva kanály obsahujú zložky rýchlosti vektora X a Y tokovej matice. Všetky rýchlosti v týchto vstupných dátach musia byť korešpondujúce k sebe veľkosťou. Náš dataset obsahuje 10058 vzoriek na tréning.

Výstupom tejto siete sú dve hodnoty, predikované zložky vektora rýchlosti do budúceho snímku videa. Štruktúra siete je 12 konvolučných vrstiev s 32 filtrami každá a 3x3 konvolúciami s ReLu po každej. Následne máme 1 poolingovú vrstvu a na konci jednu plne spojenú vrstvu, ktorá má na výstupe 2 hodnoty. Ako nákladovú funkciu sme zvolili mean square error a Adam Optimizer na tréning. Presnosť nášho modelu meriame ako sumu absolútnych hodnôt všetkých odchýlok po každej epoche tréningu.

Podarilo sa nám úspešne natrénovať sieť a otestovať ju po vysegmentovaní datasetu na 3 časti. 60% na tréning, 20% na testing a 20% na validáciu. Pôvodný test vzal 60% sekvenčných dát a tie použil na tréning. Toto ale bolo veľmi neefektívne a model podával zlé výsledky na zvyšných dátach. Pomocou náhodného vzorkovania sme zlepšili všeobecnosť trénovacej a testovacej sady a následne sme dostávali lepšie výsledky. Na úplné vyhodnotenie tohto prototypu budeme musieť ho zakomponovať do nášho trasovacieho algoritmu a odpozorovať zlepšenie výsledkov.

Ako ďalší experiment sme skúsili odstrániť posledné 2 kanály obsahujúce tokovú maticu. Tu sme chceli odpozorovať, či tieto kanály sú potrebné pre tréning siete. Pri pozorovaní tréningu a výstupov, môžeme zhodnotiť, že bez tokovej matice sa neurónová sieť nedokázala naučiť ani trénovaciu sadu.

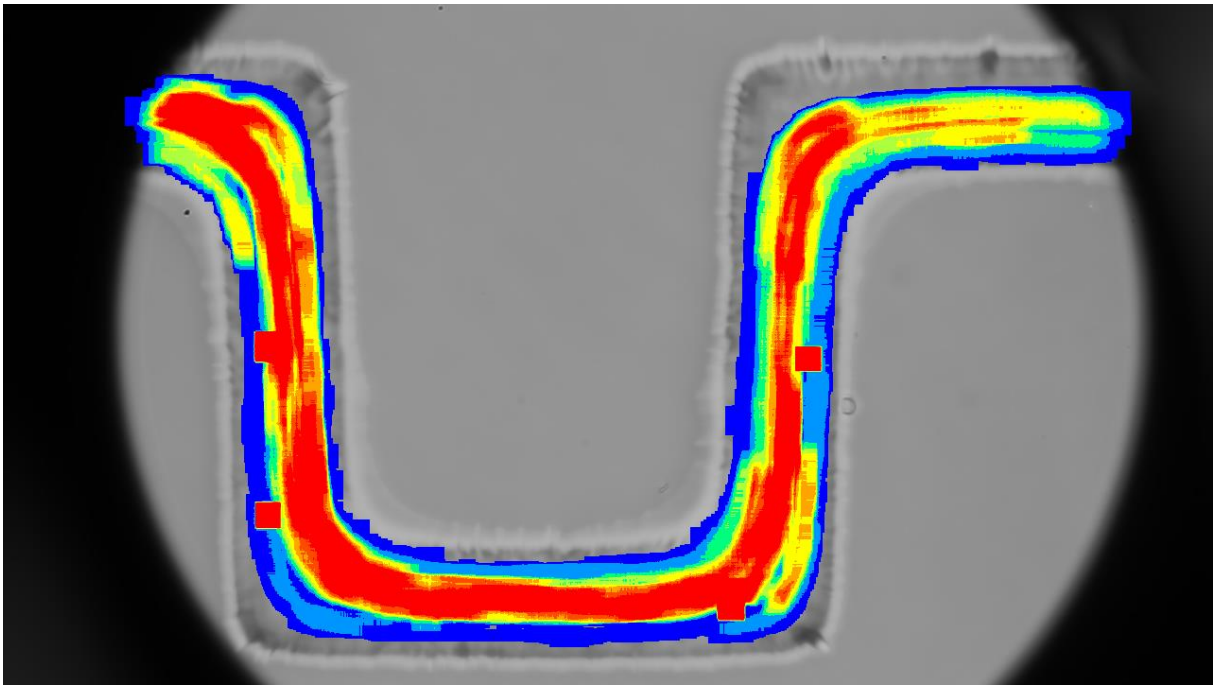
Ďalej sme sa chceli pozrieť na dopad augmentácie dát datasetu. Cieľ je mať viac dát na tréning siete. Každú vzorku datasetu sme rotovali o 90 stupňov. Taktiež sme museli upraviť vektory rýchlosti pri rotácií, a to sme zabezpečili výmenou zložiek a prenásobením výslednej zložky X -1. Ako dôsledok sa tréning spomalil o 40% ale všeobecná chyba natrénovaného modelu zostala rovnaká, s potenciálom namnoženia ojedinelých prípadov v budúcnosti.

Následne sme použili na tréning dáta červených krviniek zo simulácie. Chceli sme skúsiť tento prístup s cieľom zvýšiť rozmanitosť nášho datasetu. Takýmto spôsobom sme mali 6x množstvo pôvodných dát. Takto natrénovaný model nebol schopný sám o sebe predikovať trasovanie na našom datasete. Napriek tomu sme ešte skúsili dať dáta z datasetu dohromady s dátami zo simulácie a v tomto prípade sme dostávali rovnaké výsledky ako bez nich. To znamená, že tieto dáta nijakým spôsobom nezhoršujú natrénovaný model. Predpokladaná chyba je v nerozmanitosti simulačných dát a bude to vyžadovať ďalšie experimenty v budúcnosti.

Ako posledný kritický krok sme integrovali túto sieť do trasovacieho algoritmu. Vstupné dáta sú jednoducho dostupné na generovanie pre túto sieť a tento proces bol relatívne jednoduchý. Sieť je schopná sa pozerať do budúcnosti aj minulosti (indexy snímkov +1 a -1), keďže do minulosť stačí invertovať hodnoty vektorov. Sieť sme použili na konci trasovacieho algoritmu na finálne zlepšenie výsledkov. Podarilo sa nám znížiť priemerný počet dier na trasu z 2.3 na 1.8 čo zvýšilo priemernú dĺžku trasy na 77 snímkov. Toto samo o sebe je podstatné zlepšenie výkonu a vyzerá sľubne do budúcna.

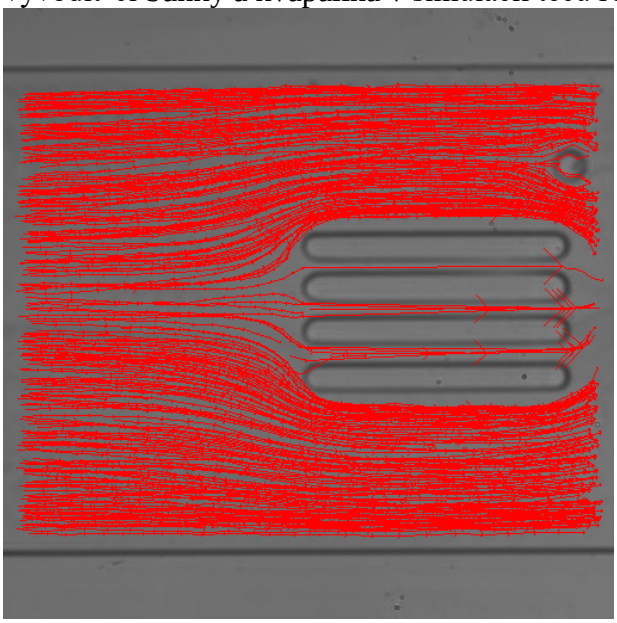
6. Validácia simulácií

Po získaní všetkých dát z videí je potrebné vyhodnotiť dáta a zostaviť metriky. Po spojení detekcií do trás, každá trasa korešponduje jednej bunke. Výstupy celého nášho spracovania obrazu sú všetky detegované obdĺžniky buniek a ich zaradenie do trás. Z tohto dokážeme vytvoriť veľa metrík na validáciu simulácií.



Prvým príkladom je heatmapa. Tá nám znázorňuje frekvenciu výskytu červených krviniek v danom mieste v kanáli. Následne ju môžeme priamo porovnávať so simuláciou.

Ďalším príkladom sú smery tokov buniek v danom videu. Z tohto môžeme následne vyvodiť či bunky a kvapalina v simulácií tečú rovnakým spôsobom. Vizualizáciu tohto toku možno vidieť na obrázku.



Dôležitým faktorom sú taktiež magnitúdy rýchlostí buniek. Oplatí sa sledovať rýchlosti v kritických oblastiach ako napríklad okolo prekážok alebo záhybov kanála. Následne je potrebné v týchto miestach priamo porovnávať dáta zo simulácie s video dátami. Do určitej miery sme taktiež schopný získať 3D informáciu o toku. Bunky v rovnakých bodoch sa pohybujú rôznymi rýchlosťami a bunky okolo okrajov kanála sa pohybujú pomalšie ako ostatné. Z toho dôvodu vieme pomôcť pri validácií simulácie sprostredkovaním rýchlostí v rôznych vrstvách toho istého bodu.

Záver

Vo výsledku automatizovaný zber dát červených krviniek z videa podáva veľmi dobré výsledky, hlavne vďaka využitiu neurónových sietí. Detekcia prvý krát dosahuje výsledky vďaka ktorým je možné používať algoritmus ako celok na zber dát bez veľkého dopadu na kvalitu výsledkov. Treba ale spomenúť že ďalšie inkrementálne vylepšenia nie sú triviálne na

experimentovanie a implementáciu. Stále je do budúcna možné rozšíriť dataset a dosiahnuť robustnejšie modely.

V prípade trasovania je momentálna cesta vytvorenie redundancií a validácií pri spájaní buniek do trás. Pridanie neurónovej siete bolo prvým krokom správnym smerom k lepším výsledkom, s možnosťou výmeny celého algoritmu za neurónovú sieť v budúcnosti.

V tejto práci sme prešli viaceré témy okolo spomínanej detekcie a trasovania. Analýza ukázala, že naše algoritmy podávajú porovnateľné výsledky inými aplikáciami týchto algoritmov v praxi v detekčnom kroku a pri trasovaní sa pomaly približujeme taktiež k perfektným výsledkom. Celkovo vývoj týchto algoritmov a nástrojov je perspektívny do budúcnosti.

Referencie

Calder, M. a. (2018). *Computational modelling for decision-making: where, why, what, who and how* (Zv. 5).

Dao, M. a. (2006). Molecularly based analysis of deformation of spectrin network and human erythrocyte. *Materials Science and Engineering C*, 1232-1244.

Domingos, P. (2012). A Few Useful Things to Know About Machine Learning. *Commun. ACM*, 78–87.

Dufour, A. a.-M. (2011). 3-D Active Meshes: Fast Discrete Deformable Models for Cell Tracking in 3-D Time-Lapse Microscopy. *Image Processing, IEEE Transactions on*, 1925 - 1937.

Fedosov, D. a. (2010). Blood flow and cell-free layer in microvessels. *Microcirculation*, 615-628.

Heikkila, S. U. (2017). Cell Tracking via Proposal Generation and Selection. *CoRR*.

I. Jančigová, C. I. (2015). A novel approach with non-uniform force allocation for area preservation in spring network models. *AIP Conference Proceedings*.

I. Jančigová, I. C. (2014). An ESPResSo implementation of elastic objects immersed in a fluid. *Computer Physics Communications*, 3.

Ka, M. a.-B.-d.-S. (2013). Segmentation and Shape Tracking of Whole Fluorescent Cells Based on the Chan-Vese Model. *IEEE transactions on medical imaging*.

Kittler, J. I. (1987). The Adaptive Hough Transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 690-698.

Kleineberg, K.-K. a. (2017). *Geometric Correlations Mitigate the Extreme Vulnerability of Multiplex Networks against Targeted Attacks* (Zv. 118). doi:10.1103/PhysRevLett.118.218301

Marton, J. J. (2017). *Optimization of periodic crew schedules with application of column generation method* (Zv. 83).

Mučka, F. (2017). Algorithms and their implementation for analysis and image processing from recordings of biological experiments. *Master thesis*.

Reyes-Aldasoro, C. a. (2007). Measuring Red Blood Cell Velocity with a Keyhole Tracking Algorithm. *11th Mediterranean Conference on Medical and Biomedical Engineering and Computing 2007*, 810-813.

Tomášiková, J. (2017). Processing and analysis of videosequences from biological experiments using special detection and tracking algorithms. *Master thesis*, 63.

Weinberger, G. H. (2016). Densely Connected Convolutional Networks. *CoRR*.

Yi, Y. W. (2017). Stem cell motion-tracking by using deep neural networks with multi-output. *Neural Computing and Applications*, 1-13.