Zpracování grafických výsledků elektroforézy virových proteinů pomocí analýzy obrazu a následného vyhodnocení v Pythonu

Pokročilé zpracování obrazů

Bc.Jakub Seiner, Bc.Jakub Žváček

2024

1 Úvod

Inspirací a podnětem pro vypracování tohoto projektu bylo to, že v našem okolí máme několik kolegů, kteří pracují v biochemických a podobně zaměřených laboratořích. Velmi běžnou laboratorní činností jsou tam analýzy pomocí gelové elektroforézy, které jsou zásadní pro určení molekulové hmotnosti zpracovávaných vzorků, což je jedním ze základních údajů o každé biomolekule. Tato procedura je tak vykonávána velmi často a proto by její zefektivnění a případná automatizace přinesly velkou časovou úsporu pro výzkumníky v těchto laboratořích.

1.1 Popis metody

Gelová elektroforéza je metodou relativní, tedy molekulová hmotnost vzorku nemůže být určena sama o sobě, ale je nutné porovnání vzorku se standardy o známých molekulových hmotnostech. Prakticky je na předem připravený gel nanesen do jedné dráhy standard, který obsahuje vzorky o definovaných molekulových hmotnostech, které se během procesu rozdělí a do vedlejších drah jsou naneseny vzorky ke stanovení. Samotné dělení probíhá díky tomu, že vzorky jsou předem zpracovány tak, aby nesly stejný záporný náboj. Následně je na gel přivedeno stejnosměrné napětí, které unáší vzorky „vzhůru“ směrem ke kladné elektrodě. Čím má vzorek menší molekulovou hmotnost, tím snáze putuje skrze póry v gelu a je tak unášen více než vzorky s vysokou molekulovou hmotností. Díky tomuto dojde na rozdělení na charakteristické proužky podle molekulových hmotností. Pro to, aby mohly být výsledky vizualizovány je nutné obarvení celého gelu s nanesenými vzorky v barvivu Coomassie blue, což vysvětluje také modravé zabarvení původních nezpracovaných dat.

1.2 Existující řešení

Každý gel je po dokončení elektroforetického cyklu nejdříve vyfocen a až následně jsou z něj zpracovávána data, ovšem dosavadní praxe je taková, že jsou tato data vyhodnocována ručně. To je však neefektivní a zdlouhavé a proto jsme přišli s myšlenkou, zda by tento proces nešlo automatizovat. Na základě drobné rešerše jsme zjistili, že existuje řada programů, které jsou této analýzy schopny, například produkty společností Bio-Rad nebo Syngene, avšak většina z nich vyžaduje nákup licence, která může být nákladná a pohybuje se v závislosti na konkrétním programu a jeho verzi v řádech tisícovek až desítek tisíc korun. Mimo to existují také volně dostupné alternativy, například Scion Image, což je program pro analýzu nejen gelové elektroforézy ale také dalších analytických metod používaných v biochemii. Velmi podobně ovládaný plugin obsahuje také program FIJI pro zpracování obrazových dat. Nevýhodou obou zmíněných volně dostupných programů je to, že většina činností není automatizovaná a jednotlivé píky a proužky vzorků musí být stále vybírány ručně. Základní myšlenkou bylo vytvořit algoritmus, který by byl schopen rozpoznat jednotlivé vzorky na gelu a v porovnání se standardem by konečným výstupem byla molekulová hmotnost vyhodnocovaných vzorků, případně interval molekulových hmotností.

Cílem byla také maximální možná míra automatizace, aby byl celý postup co nejméně náročný jak na čas, tak na znalosti a počítačové schopnosti obsluhy. Vstupními daty do projektu byla již existující obrazová data z laboratoří, celkem jde asi o 20 obrázků s různou kvalitou výsledků. Za kvalitní výsledky gelové elektroforézy lze považovat jasně rozdělené proužky, které jsou co nejužší a zároveň velmi kontrastní oproti samotnému gelu. Nekvalitní výsledek je pak takový, kde je vzorek rozmytý přes velkou část své dráhy. Poté nelze spolehlivě určit jeho molekulovou hmotnost, protože proužek vzorku může být v rozsahu tisícovek Daltonů (1 Da = 1 g/mol, jednotka molekulové hmotnosti používaná v biochemii) a má téměř nulovou vypovídající hodnotu a stejně tak nelze příliš mnoho údajů určit ani z obrazové analýzy takovéhoto vzorku.

2 Metodika

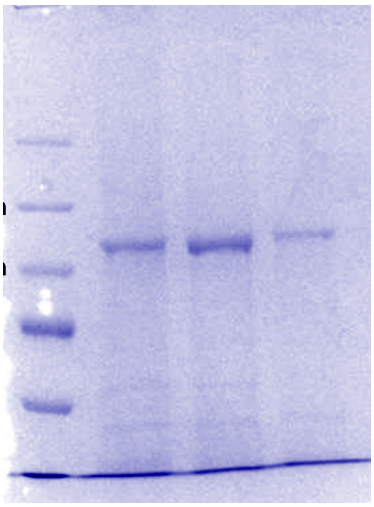
2.1 Předzpracování

První akcí se všemi obrázky byla selekce těch s kvalitními výsledky, které jsou vhodné pro návrh celého algoritmu, respektive pro to, aby algoritmus dával věrohodné výsledky. Zvolené obrázky bylo nutné nejdříve oříznout tak, aby obsahovaly pouze gel s nanesenými vzorky. Celé fotografie totiž obsahují také okraje gelu, které tvoří vizuálně významné hrany, na které by se mohly následné algoritmy zbytečně zaměřovat, nebo i popisky, které jsou sice důležité pro laboratorní praxi a orientaci, avšak pro počítačové zpracování nenesou žádné podstatné informace. Další nezbytnou operací bylo převedení do šedotónového spektra pro snížení dimenzionality celého problému (Obr. **2**), zároveň však neztrácíme žádnou podstatnou informaci o datech, jelikož ani původní data neobsahují informace, které by byly neseny různobarevným spektrem.

2.2 Použité algoritmy pro zpracování

Pro zpracování obrazových dat byl zvolen program FIJI včetně rozšíření CLIJ. Nejprve byl aplikován filtr šumu pro redukci náhodného šumu. Jako dva nejvhodnější typy filtrů se ukázaly Gaussian blur a Mean filter. Gaussian blur vypočítává novou hodnotu každého pixelu pomocí váženého průměru okolních pixelů, přičemž váhy jsou pixelům přikládány podle jejich rozložení v prostoru ve smyslu Gaussovy funkce, tedy s rostoucí vzdáleností od vypočítávaného pixelu klesá váha daných bodů. Naproti tomu Mean filter nahrazuje hodnotu každého pixelu průměrem hodnot okolních pixelů. Zatímco Gaussian Blur je vhodnější pro obrazová data s Gaussovským šumem, Mean filtr je výhodnější pro obrazy s náhodným šumem typu salt and pepper. S ohledem na histogram obrázku, který vykreslil typickou křivku Gaussova typu, což poukazuje spíše na Gaussovský šum, byl jako filtr zvolen právě Gaussian Blur, konkrétně s parametry x, y a $z = 2.0$ (Obr. **3**).

Následující úpravou je binarizace, což je proces, během kterého dochází k převodu na černobílý formát obrázku, tedy každému pixelu je přiřazena hodnota



Obr. 1: Neupravený originál obrázku

0 nebo 1. Jako lepší pro naše data se ukázaly metody lokálního prahování, které aplikují různé hodnoty thresholdu na různé oblasti obrázku, kdežto metody globální aplikují stejný threshold na celý obrázek. Nejvhodnější pro náš účel byl Local Threshold Mean, který dával při úpravě parametrů velmi dobré výsledky. Konkrétně byly parametry nastaveny na hodnoty *radius* = 50.0 a *c-value* = 6.0 (Obr. **4**).

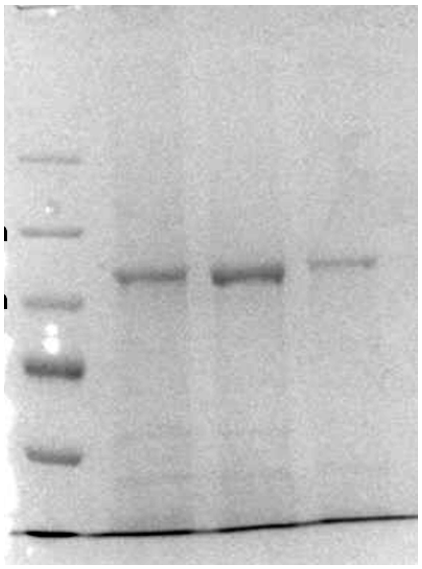
Vzniklý binární obrázek neobsahoval téměř žádné rušivé signály, které by pro stanovení byly nežádoucí, proto jsme mohli provést dilataci útvarů tak, aby došlo k uzavření a spojení všech dílčích útvarů. Fakticky však tato dilatace musela být udělána pomocí příkazu Erode Box, jelikož jsme v inverzním módu oproti typické práci a zajímají nás černé, nikoliv bílé oblasti (Obr. **5**).

Následně byl nastaven Threshold obrázku tak, aby se inverze opravila, tedy hledané útvary na obrázku jsou nyní bílé a pozadí je černé (Obr. **6**). Tento krok byl nezbytný pro pokračování zpracování, konkrétně pro labeling.

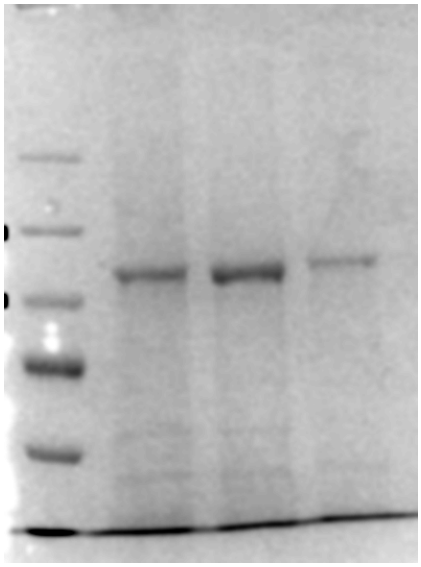
Pro labeling, tedy označení jednotlivých útvarů na obrázku byl použit vestavěný nástroj Analze particles. Abychom se vyhnuli počítání případných fragmentů nebo nečistot v okolí okrajů stejně jako dlouhé základní čáry v úplně dolní části, nastavili jsme parametr particle size na interval 250-1500 (Obr. **7**). Díky tomu započítáváme jen pro nás relevantní částice. Nejdůležitější hodnotou, kterou jsme tímto získali jsou souřadnice X a Y jednotlivých útvarů, které slouží jako vstupní data do dalších kroků analýzy.

2.3 Použité algoritmy pro zpracování

Pro další zpracování byl použit jazyk Python. Python je vysokoúrovňový, interpretovaný programovací jazyk, který se těší velké popularitě mezi programátory



Obr. 2: Obrázek v šedotónovém spektru



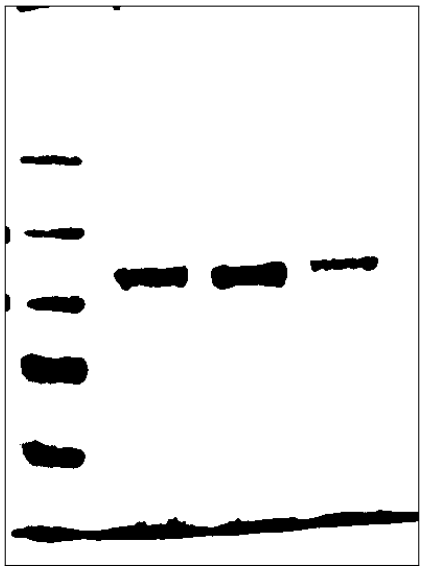
Obr. 3: Obrázek s aplikovaným Gaussian blurem

po celém světě. Vytvořen Guidem van Rossumem a poprvé uveden v roce 1991.

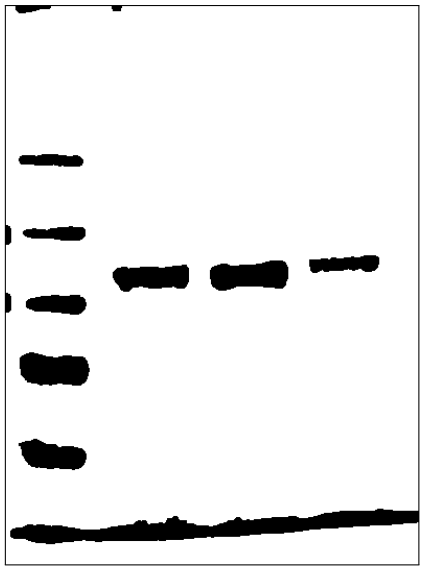
Jedním z hlavních důvodů, proč je Python tolik oblíbený, je jeho srozumitelná a čitelná syntaxe. Díky ní je Python snadno naučitelný, což ho činí ideálním nástrojem pro začínající programátory. Syntaxe Pythonu je navržena tak, aby byla intuitivní, a umožňuje psaní kódu, který je snadno čitelný a udržovatelný. To je velkou výhodou v týmových projektech, kde je přehlednost kódu klíčová.

Python je interpretovaný jazyk, což znamená, že kód je vykonáván přímo interpretem bez potřeby kompilace. To usnadňuje vývoj a testování programů, protože změny v kódu mohou být rychle provedeny a okamžitě testovány. Díky těmto vlastnostem se Python stal nástrojem volby nejen pro webové aplikace, ale také pro vědecké výpočty, datovou analýzu, umělou inteligenci, automatizaci systémových úloh a mnoho dalších oblastí.

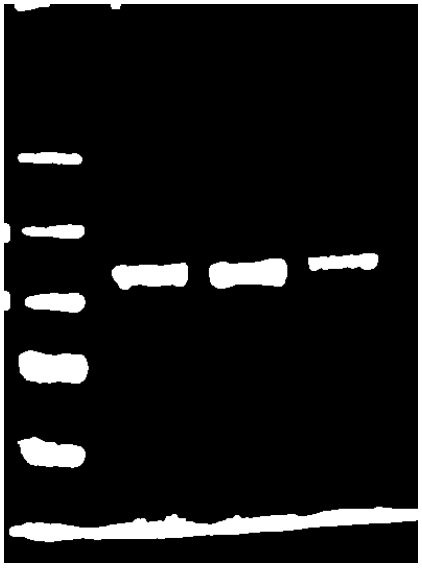
Výstupem z výše zmíněných metod zpracování byl textový soubor s několika parametry souvisejících s nasegmentovanými proužky. Nejdůležitějšími parametry jsou sloupce x a y souřadnic. V jazyce Python byl tento textový soubor rozdělen a sloupce s důležitými daty separovány. Dále byl vytvořen kód který seřadí proužky z analýzy podle polohy, oddělí standard kterému pro nejkrajněji umístěné látky přiřadí zadané molekulové hmotnosti. Dále se ve scriptu vytvoří lineární model který jako vstup používá souřadnici y a jako target molekulovou hmotnost a po natrénování dopočítá molekulové hmotnosti i vzorkům, podle jejich souřadnic a výsledky vypíše do konzole.



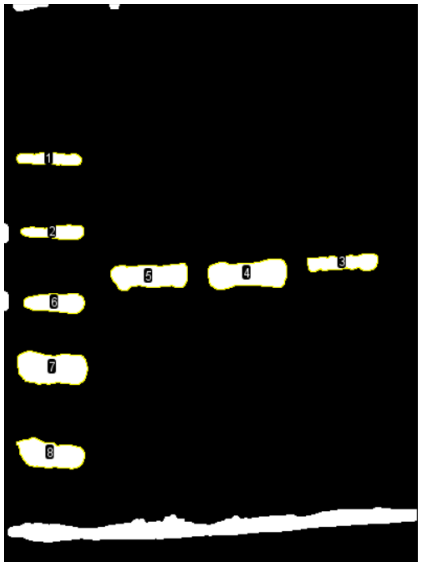
Obr. 4: Binarizovaná verze obrázku



Obr. 5: Dilatace černých oblastí



Obr. 6: Upravení thresholdu obrázku



Obr. 7: Označené částice pro další analýzu

```python
import numpy as np
from sklearn.linear_model import LinearRegression

results = []
y_targ = []
targets = []
order_list = []
inputs = []
disp_res = []
masses = [100, 500]

with open('Results_3004.txt', 'r') as file:
    next(file)
    for line in file:
        values = line.strip().split('\t')

        y_targ.append(round(float(values[3])))
        results.append(round(float(values[4])))
    base = min(y_targ)
    high_base = base + 0.1 * base
    low_base = base - 0.1 * base

with open('Results_3004.txt', 'r') as file:
    next(file)
    for line in file:
        values = line.strip().split('\t')
        if round(float(values[3])) < high_base and round(float(values[3])) > low_base:
            order_list.append(round(float(values[4])))

    max_std = max(order_list)
    min_std = min(order_list)

with open('Results_3004.txt', 'r') as file:
    next(file)
    for line in file:
        values = line.strip().split('\t')
        if round(float(values[4])) == max_std:
            targets.append(round(float(values[4])))
            inputs.append(round(float(masses[1])))
        elif round(float(values[4])) == min_std:
            targets.append(round(float(values[4])))
            inputs.append(round(float(masses[0])))

# model fit
x = np.array(inputs).reshape(-1, 1)
y = np.array(targets).reshape(-1, 1)

regr = LinearRegression()
regr.fit(y, x)

pred_mass = regr.predict(np.array(results).reshape(-1,1))
pred_mass = pred_mass.reshape(1, -1)
for i in range(len(results)):
    disp_res.append([round(pred_mass[0][i], 0), results[i]])

print(disp_res)

```

Obr. 8: Ukázka scriptu pro separaci souřadnic

Results_3004 – Poznámkový blok

Soubor Úpravy Formát Zobrazení ° Nápověda

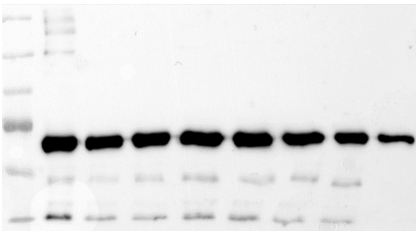
	Area	Mean	X	Y	XM	YM	Perim.	Slice
1	328	255	45.564	148.284	45.564	148.284	135.740	1
2	468	255	47.293	218.359	47.293	218.359	131.012	1
3	461	255	324.331	247.379	324.331	247.379	155.640	1
4	974	255	234.297	259.147	234.297	259.147	181.539	1
5	754	255	140.798	260.182	140.798	260.182	172.711	1
6	563	255	48.649	286.631	48.649	286.631	137.640	1
7	958	255	47.729	348.882	47.729	348.882	157.054	1
8	692	255	47.504	433.027	47.504	433.027	138.326	1

Obr. 9: Formát textového souboru s výsledky pro zpracování v Pythonu

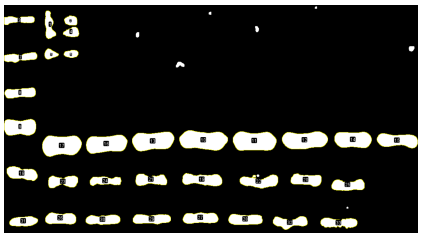
3 Výsledky

Nastavení procesu zpracování v programu FIJI umožnilo získat z předem nezpracovaných dat souřadnice jednotlivých látek ve vzorku pro snímky, které byly pořízené pro různé vzorky za různých podmínek. Pro tento účel byl proces navrhován tak, aby byl co nejunivevrzálnější a v jednom nastavení dokázal zpracovat různé snímky. Díky tomu se však také mírně liší kvalita výstupů, jak je patrné na obrázcích níže.

Navzdory tomu, že samotný snímek (Obr. **10**) je poměrně silně přesvětlen a špatně zaostřen, připravený algoritmus dokázal úspěšně segmentovat všechny významné oblasti z původního snímku, avšak byl upraven parametr particle size, protože všechny útvary na snímku byly obecně mnohem větší (Obr. **11**).

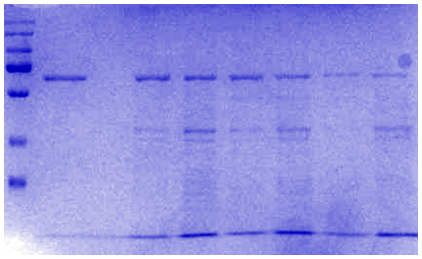


Obr. 10: Originální snímek

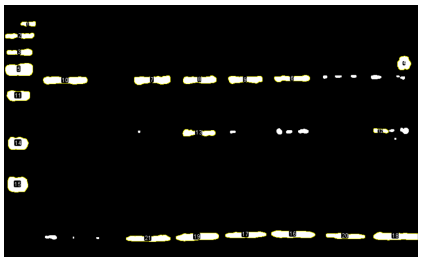


Obr. 11: Zpracovaný snímek

U druhé dvojice obrázků je zřejmé (Obr. **12** a Obr. **13**), že díky příliš univerzálnímu nastavení se nepodařilo zachytit všechny proužky, problematické jsou zejména ty s nižším kontrastem vůči okolí, které jsou hůře viditelné i lidským okem. Mimo to je problémem také nečistota v pravé horní části, která je velmi kontrastní a proto ji program interpretuje jako součást vzorku k identifikaci.



Obr. 12: Originální snímek



Obr. 13: Zpracovaný snímek

4 Diskuze

Aplikací předepsaného postupu zpracování obrázku jsme pro různá vstupní data získali velmi odlišné výsledky. Jejich kvalita byla determinována hlavně kvalitou vstupních dat, která se však velmi liší a díky tomu nelze v tomto případě mluvit o 100% spolehlivosti této metody zpracování obrazu. Optimální segmentace každého jednotlivého obrazového vstupu by zřejmě bylo možné dosáhnout, avšak jednotlivé postupy by se mezi sebou lišily hodnotami parametrů aplikovaných metod, některé krajní případy možná i samotnou metodou, například binarizace. Již v tento moment pak byly pro různé obrázky použity jiné meze v příkazu analyze particles tak, aby docházelo jen k označování těch částic a útvarů, které chceme skutečně studovat a analyzovat. Z našeho pohledu je největším kamenem úrazu nekonzistence v pořizování obrazových záznamů, díky čemuž mohou záběry být rozostřené, nedostatečně osvětlené, nebo naopak příliš přesvícené. V důsledku toho je pak velmi obtížné sestavit univerzální algoritmus, který by spolehlivě fungoval. Samotný vzorek stejně jako optická soustava mohou obsahovat nečistoty, které pak zkreslují výsledky, protože nečistoty v gelu mohou vypadat velmi podobně jako sledované útvary a nečistoty usazené na čočce mohou za určitých okolností rovněž připomínat sledované částice a tím narušit jejich sledování a automatickou analýzu.

5 Závěr

Práce si vytyčila za cíl sestavení algoritmu, který by byl schopen extrakce molekulových hmotností přímo z obrazových dat pořízených v laboratoři. Připravený algoritmus byl následně testován na dalších typech dat pro ověření jeho spolehlivosti, ta však kolísala v závislosti na kvalitě vstupních snímků. Vzhledem k zamýšlené univerzalitě procesu byly problémem detaily některých snímků, které zůstaly mimo rozlišovací schopnost programu. Jako společný prvek špatně identifikovaných proužků ve vzorcích by se dal označit nedostatečný kontrast vůči pozadí, některé proužky jsou špatně viditelné i lidským okem. Zde může jít o problém přímo dané metody, kde jej principielně nelze odstranit, případně může jít o problém se záznamovou technikou. V tomto případě lze navrhnout úpravy a zlepšení kvality záznamů, díky čemuž by bylo možné sestavit spolehlivější algo-

ritmy. Podnětem pro další zlepšení by pak mohlo být také sestavení série několika různých algoritmů, které by měly vymezeny oblasti použití. Například pro optimální snímky by byl sestaven jeden, pro nedostatečně osvětlené snímky jiný atp. Další možností dalšího postupu by bylo také použití neuronové sítě, avšak zde je významným limitem malá velikost datasetu, jelikož dat jsou aktuálně dostupné pouze nízké desítky. Pokud však budou výstupy elektroforéz homogenní kvality, bylo by použití tohoto postupu možné a umožnilo by časovou úsporu při zpracování tohoto typu obrazových dat.