Wykrywanie i śledzenie komórek glistopodobnych w programie ImageJ

(Worm-like cell detection and tracking in ImageJ)

Artur Rosa

Praca magisterska

Promotor: dr Andrzej Łukaszewski

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

24 lipca 2020

Streszczenie

Polskie streszczenie

English abstract

Spis treści

Rozdział 1.

Wprowadzenie

1.1. Wstęp

...

1.2. Background

1.2.1. Jakich danych i do czego potrzebują biolodzy

...

1.2.2. Pozyskiwanie danych

...

Rozdział 2.

Wykrywanie i śledzenie komórek

2.1. Opis problemu

2.1.1. Obrazy wejściowe

...

2.1.2. Pożądany efekt

...

2.2. Powiązane prace

...

2.3. Wykrywanie komórek

2.3.1. Wstęp

Problem opisany w sekcji 2.1. zdefiniowany jest dla nagrań spod mikroskopu. Postanowiłem jednak najpierw rozwiązać podobny problem, ale zdefiniowany dla pojedynczego obrazu. Rozwiązanie tego problemu mogłoby z łatwością zostać uogólnione na stos obrazów (nagranie). W tym rozdziale opiszę rozwiązanie uproszczonego problemu: oznaczanie komórek widocznych na pojedynczym obrazie. Przez „oznaczenie komórki" mam na myśli odnalezienie łamanej przechodzącej przez środek komórki, a więc jej kręgosłupa.

2.3.2. Dane wejściowe

Oznaczenie wszystkich komórek widocznych na obrazie można rozłożyć na dwa osobne problemy:

1. Określenie liczby oraz lokalizacji poszczególnych komórek
2. Odnalezienie kształtu poszczególnych komórek.

W niniejszej pracy zdecydowałem się nie rozwiązywać automatycznie pierwszego problemu. Zamiast tego użytkownik zobowiązany jest ręcznie zaznaczyć dokładnie jeden punkt wewnątrz każdej komórki widocznej na obrazie. Wymóg ten dotyczy tylko pierwszej klatki nagrania, co opiszę dokładniej w dalszej części pracy (2.4.).

Danymi wejściowymi są zatem obraz $\mathbf{I}$ oraz zbiór punktów $\mathbf{P}$ lokalizujących komórki.

2.3.3. Wybór kanału i wstępne przetwarzanie obrazu

Obraz wejściowy składa się z dwóch podstawowych kanałów (2.1.1.). Chcąc jak najdokładniej oznaczyć początek i koniec komórki, a także miejsca ich podziału, postanowiłem wybrać kanał, który zawiera wyraźną informację o krawędziach w tych miejscach. O ile kanał z fluorescencją mógłby bardzo dobrze sprawdzić się do określenie liczby oraz lokalizacji poszczególnych komórek (w przypadku ich niewielkiej liczby), o tyle drugi kanał zawiera dużo dokładniejszą informację na temat krawędzi komórek.

Celem wstępnego przetwarzania obrazu wejściowego jest w tym przypadku oddzielenie poszczególnych komórek od otoczenia.

Shape index map

...

2.3.4. Szkieletyzacja i wstępna detekcja komórek

Zgodnie z powyższą obserwacją, możemy łatwo oddzielić komórkę od jej otoczenia na obrazie ustalając pewien próg $t \approx 0$. Załóżmy przez chwilę, że binaryzując w ten sposób mapę indeksów kształtu otrzymamy obraz $\mathbf{I}_{bin}$, na którym każdy piksel leżący wewnątrz dowolnej komórki będzie miał wartość 1, natomiast każdy piksel należący do zewnętrznego obrysu dowolnej komórki (nie należący do komórki, lecz sąsiadujący z pikselem należącym do niej) będzie miał wartość 0. Przy takim założeniu każda komórka jest niezależną „wyspą" na binarnym obrazie $\mathbf{I}_{bin}$. Chcąc odnaleźć „kręgosłup" komórki (łamaną przechodzącą przez jej środek) chcemy tak naprawdę

znaleźć łamaną, która jest równoodległa od jej krawędzi. Bardzo podobny problem rozwiązują algorytmy do wyznaczania szkieletu – ich celem jest odnalezienie zbioru punktów równoodległych od co najmniej dwóch brzegów.

Na potrzeby tej pracy do szkieletonizacji użyta została implementacja algorytmu „3D thinning algorithm"[1] w formie pluginu dla programu ImageJ[2]. Mimo iż wtyczka pozwala na szkieletyzację obrazów 3D, w tym przypadku została użyta do przetworzenia pojedynczego obrazu 2D. Wynikiem szkieletyzacji jest binarny obraz o pewnych właściwościach. Każdy aktywny piksel można przyporządkować do trzech grup:

- końcówki – mają mniej niż 2 sąsiadujące aktywne piksele
- węzły – mają więcej niż 2 sąsiadujące aktywne piksele
- połączenia – mają dokładnie 2 sąsiadujące aktywne piksele.

Przedstawiając szkielet jako graf końcówki tworzyłyby wierzchołki o stopniu równym 1 lub 0, wierzchołki o większych stopniach przedstawiałyby zbiory sąsiadujących ze sobą węzłów, natomiast krawędzie reprezentowałyby zbiory sąsiadujących ze sobą połączeń (zakończonych zbiorem węzłów lub końcówką). Taką reprezentację grafową można uzyskać za pomocą kolejnej wtyczki dla programu ImageJ tego samego autora[3]. Poza standardowymi informacjami wierzchołki i krawędzie utworzonego za jej pomocą grafu przechowują zbiory pikseli które reprezentują.

Ze względu na specyficzny kształt komórki można zaobserwować następującą właściwość: po przeprowadzeniu szkieletyzacji obrazu $\mathbf{I}_{bin}$, o ile przyjęte wcześniej założenie jest spełnione, szkielet każdej z komórek składa się dokładnie z dwóch końcówek i połączeń między nimi. Łamaną opisującą zbiór połączeń można z powodzeniem nazwać „kręgosłupem komórki". Mając do dyspozycji zbiór punktów $\mathbf{P}$ lokalizujących komórki można teraz w łatwy sposób odnaleźć dla każdej z nich graf ją opisujący (zawierający dwa wierzchołki i jedną krawędź). Jednym ze sposobów może być wyszukanie dla każdego punktu ze zbioru $\mathbf{P}$ krawędź która znajduje się najbliżej tego punktu, gdzie odległość między punktem a krawędzią zdefiniowana jest jako odległość między punktem, a najbliższym pikselem, który należy do zbioru opisywanego przez tę krawędź.

2.3.5. Wybór krawędzi w węzłach

Niestety przyjęte założenie o tym, że każdy piksel należący do zewnętrznego obrysu dowolnej komórki będzie miał wartość 0, nie zawsze jest spełnione. W realistycznym scenariuszu zdarza się, że jedna z końcówek komórki znajdują się na tyle blisko innej komórki, że wstępne przetwarzanie i progowanie obrazu nie powoduje ich rozdzielenia na obrazie binarnym. Czasem artefakty widoczne na obrazie wejściowym powodują, że wyspa na obrazie binarnym zawiera nie tylko komórkę, ale

także fragment innego kształtu. W zdecydowanej większości takich przypadków kręgosłup komórki można opisać spójnym podgrafem o stopniu 2 grafu reprezentującego szkielet wyspy zawierającej komórki.

Wstępna detekcja kręgosłupa danej komórki polega, tak jak w scenariuszu optymistycznym, na odnalezieniu krawędzi $\{v_0, u_0\}$ w grafie $\mathbf{G}$ leżącej najbliżej punktu opisującego komórkę, a następnie na stworzeniu z niej i jej wierzchołków nowego grafu $\mathbf{S}$. Tak utworzony kręgosłup rozszerzany jest później zgodnie z następującym algorytmem:

for each $v \leftarrow v_0, u_0$ **do**
  loop
    $\mathrm{E}_v \leftarrow$ podzbiór krawędzi grafu $E(\mathbf{G}) - E(\mathbf{S})$ incydentnych do v,
        które nie tworzą cyklu w grafie $\mathbf{S}$
    if $\mathrm{E}_v = \emptyset$ **then**
      break
    end if
    $\{v, u\} \leftarrow \mathbf{argmax}_{e \in \mathrm{E}_v} q_v(e)$
    $\mathbf{S} \leftarrow \mathbf{S} \cup (\{u\}, \{\{v, u\}\})$
    $v \leftarrow u$
  end loop
end for

gdzie funkcja $q_v(e)$ jest funkcją oceny, która służy do wyboru najmocniej związanej krawędzi. Siła wiązania zdefiniowana jest tutaj jako najniższa wartość indeksu kształtu (oryginalnego obrazu) dla piksela leżącego na krawędzi e, w okolicy wierzchołka v (w odległości nie większej niż pewna stała d od środka masy pikseli tworzących węzeł v).

Takie rozwiązanie sprawdza się dobrze dla komórek, których kręgosłup znajduje się w grafie $\mathbf{G}$, oraz jego zakończenia w tym grafie mają stopień 1. Pomijam na ten moment przypadek, gdy szukany kręgosłup komórki nie istnieje w grafie $\mathbf{G}$. W pozostałych przypadkach przynajmniej jedno z zakończeń szukanego kręgosłupa $\mathbf{S}$ ma w grafie $\mathbf{G}$ stopień większy niż 1. Zatem, o ile funkcja oceny $q_v(e)$ sprawdzi się dobrze jeśli chodzi o dobór krawędzi należących do szukanego kręgosłupa komórki, uzyskany na końcu algorytmu kręgosłup $\mathbf{S}'$ będzie nadgrafem szukanego kręgosłupa $\mathbf{S}$. W takim przypadku „nadmiarowa" część kręgosłupa należy do szkieletu innej komórki lub jest wynikiem artefaktu widocznego na oryginalnym obrazie. Drugi przypadek pozostawmy do ręcznego rozwiązania użytkownikowi. Pierwszy natomiast, nachodzące na siebie kręgosłupy komórek, rozwiązujemy automatycznie.

2.3.6. Rozwiązywanie konfliktów

...

2.3.7. Uzyskiwanie linii łamanej

...

2.3.8. Korekta końcówek

...

2.4. Śledzenie komórek w czasie

...

2.5. Interakcja ze strony użytkownika

...

Rozdział 3.

Opis implementacji

3.1. Kod źródłowy

...

3.2. Kompilacja i uruchomienie

...

3.3. Struktury danych

...

3.4. Architektura

...

3.5. Opis wykorzystanego API ImageJ

...

3.6. Błędy w implementacji ImageJ i sposoby na ich obejście

...

3.7. Opis sposobu dalszego rozwoju, interfejsy

...

Rozdział 4.

Zakończenie

4.1. Podsumowanie

...

4.2. Ograniczenia wynikające z zastosowanych metod

...

4.3. Dalszy rozwój

...

Bibliografia

[1] Ta-Chih Lee, Rangasami L. Kashyap, Chong-Nam Chu, *Building skeleton models via 3-D medial surface/axis thinning algorithms*, Computer Vision, Graphics, and Image Processing, 56(6):462–478, 1994.

[2] Ignacio Arganda-Carreras, *Skeletonize3D*, 2.1.1, 2017, `https://imagej.net/Skeletonize3D`.

[3] Ignacio Arganda-Carreras, *AnalyzeSkeleton*, 3.3.0, 2018, `https://imagej.net/AnalyzeSkeleton`.