Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Броење на јадра во слика со Watershed алгоритам

Предмет: Обработка на слика

Ментор:

Д-р Ивица Димитровски

Изработено од:

Јана Николовска 171040

Содржина

Вовед

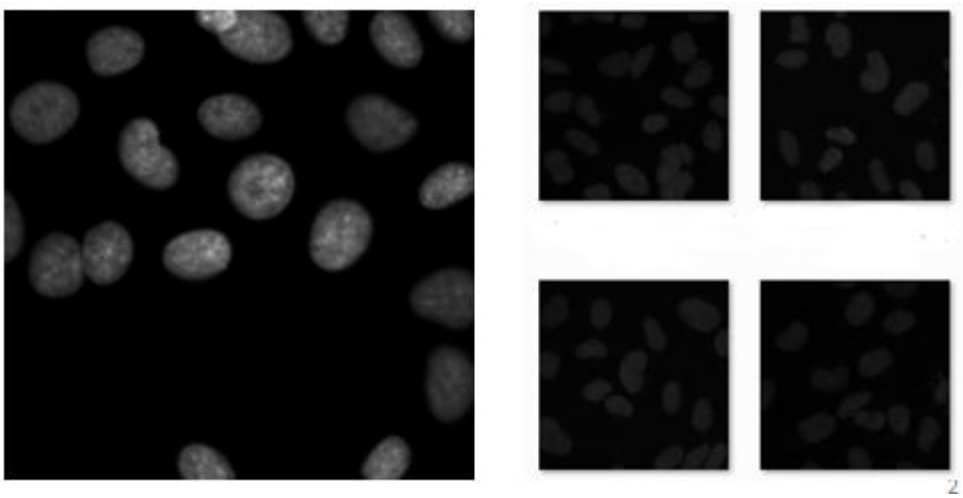
Во делот од науката која го покрива процесирањето на слики, watershed претставува трансформација на една црно-бела слика која вообичаено се користи за решавање на проблемот сегментирање на слика. Алгоритмите за сегментирање имаат задача сликата да ја поделат на повеќе делови или сегменти и да додели лабела на секој од пикселите така што пикселите со исти лабели би спаѓале во еден сегмент и би делеле исти карактеристики. Со watershed алгоритамот не само што можеме да ги пронајдеме сегментите, туку можеме и да ги изброиме и оваа модификација на алгоритмот наоѓа примена во пребројување на клетки, честитчки, пари итн. Во продолжение во детали ќе ја разлгедаме имплементацијата на овој код во проблемот на пребројување јадра.

Watershed алгоритам

Концептот бил претставен во 1978 година од страна на Digabel и Lantuejoul, но најголема примена била постигната во 1991 од Vincent и Soille откако претставиле нова верзија на истиот алгоритам со подобрени временски перформанси како и прецизност. Името watershed доаѓа од истоименот англиски збор кој го преведуваме како вододелница - граница на два слива во облик на линија од која водата се слева во два правца. Вообичаено оваа граница претставува планина или планинаска верига, така што поради разликата во надморска височина не постои никакво поврзување на водата што се наоѓа од едната со водата што се наоѓа од другата страна на оваа вододелница. Алгоритамот има за цел да ја трансформира црно-белата слика во нешто налик тополошка мапа поради која и го има добиено ова име. Во мапата добиена по оваа трансформација светлината ја одредува висината (колку посветла толку повисоко), а линиите составени од највисоките точки се сметаат за вододелница.

Имплементација на алгоритамот

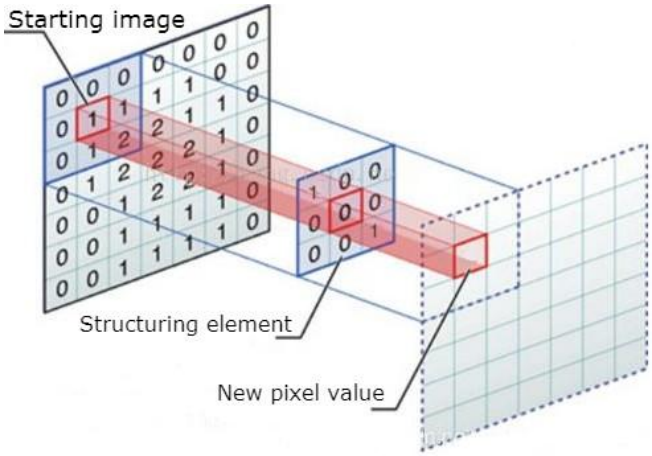
За потребите на овој проект watershed алгоритамот е имплементиран во програмскиот јазик python. Дополнително на пример сликата (слика 1), податочното множество е надополнето со слики кои ги делат истите карактеристики како и примерот (иста позадина, иста големина на јадрата) превземени од Data Science Bowl 2018[1] организиран од Kaggle (слика 1). Сите слики се претпроцесирани пред да се изврши алгоритамот и резултатите на пребројувањата се проследени со слика на која се означени јадрата.



слика 1 Пример слика, Слики од податочното множество на Kaggle

Морфолошки операции

Пред да преминеме на алгоритамот, во делот за претпроцесирање на сликите користиме морфолошки операции за измена на истите. Морфолошко процесирање на сликите е колекција од една или повеќе нелинеарни операции кои влијаат на морфологијата на сликата. Овие операции најголема примена наоѓаат во обработка на бинарни слики бидејќи не зависат



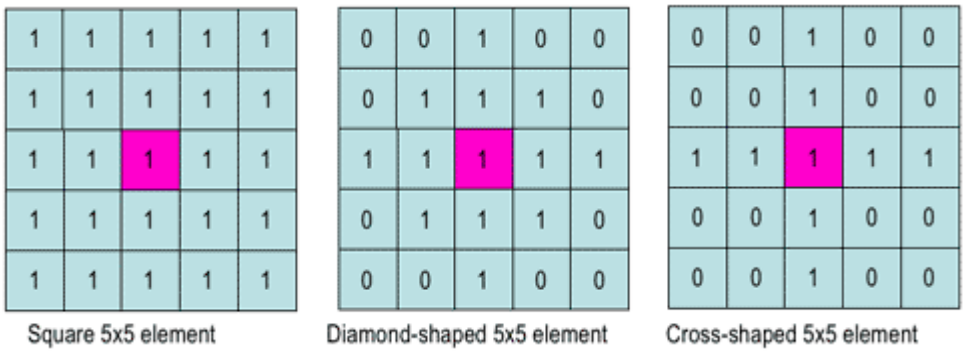
слика 2 Морфолошка операција. "Лизгање" на елементот за структурирање по димензиите на сликата

многу од вредноста на пикселот туку од неговото подредување. Постојат повеќе морфолошки трансформации кои се разликуваат меѓусебе по нивниот елемент за структурирање и логичките операции кои применуваат. (слика 2)

Structuring element

Основа на сите морфолошки операции е елементот кој структурира (structuring element) или уште познат како јадро на морфолошката операција. Овој елемент претставува бинарна матрица чии димензии и облик можеме да ги дефинираме и кој потоа се користи за трансормација на сликата која се процесира со “лизгање” на истиот тој елемент по должината и ширината на сликата и вршење на логички операции на пикселите кои ги изминува.

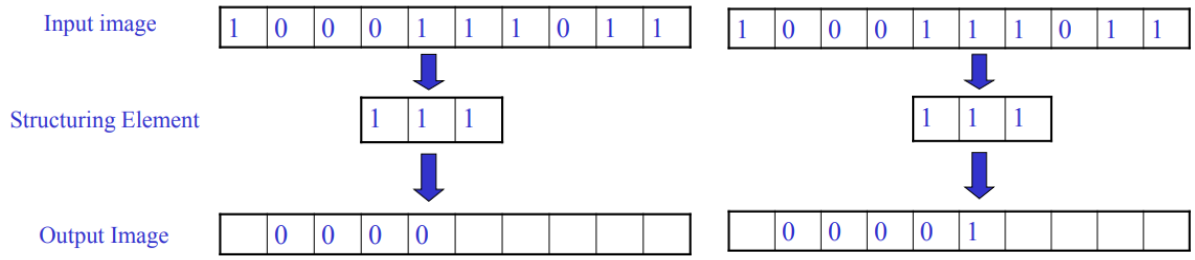
Елементот може да биде дефиниран мануелно или со функцијата getStructuringElement од библиотеката OpenCV каде може да завземе облик на правоаголник, елипса или крст. (слика 3)



слика 3 Облик на елемент за структурирање, квадрат, дијамант уште наречено елипса, крст

Ерозија

Ерозија е морфолошка операција во која вредноста на пикселот после трансформацијата е 1 доколку подматрицата од оригиналната слика целосно се совпаѓа со елементот за структурирање (логичко и).



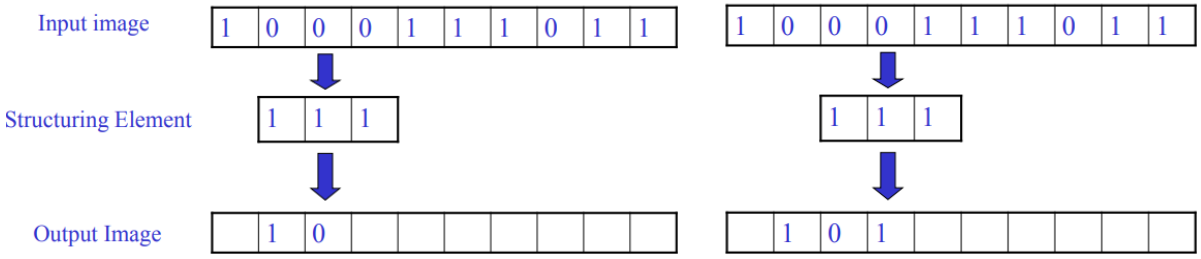
слика 4 Пример за ерозија. Еднодимензионална репрезентација на слика и еднодимензионална репрезентација на елемент за структурирање.

Ерозијата ја смалува големината на елементите на сликата така што отстранува слој на пиксели од границите на елементите. Празнините во елементите и растојанијата помеѓу елементите се зголемуваат, а малите детали целосно се губат.

слика 5 Ерозија со елипсовиден елемент за структурирање во големина 3x3 - нормална слика, трансформираната слика по 5 итерации, трансформираната слика по 10 итерации

Дилатација

Дилатација е морфолошка операција во која вредноста на пикселот после трансформацијата е 1 доколку подматрицата од оригиналната слика се совпаѓа со барем еден пиксел од елементот за структурирање. (логичко или)



слика 6 Пример за ерозија. Еднодимензионална репрезентација на слика и еднодимензионална репрезентација на елемент за структурирање.

Дилатацијата ја зголемува големината на елементите на сметка на позадината. Празнините во елементите и растојанијата помеѓу елементите се пополнуваат.

слика 7 Дилатација со елипсовиден елемент за структурирање во големина 3x3 - нормална слика, трансформираната слика по 5 итерации, трансформираната слика по 10 итерации

Opening

Opening е морфолошка операција која е комбинација од други две операции поточно ерозија проследена со дилатација. Оваа морфолошка операција се нарекува opening или отварање затоа што најпрво со ерозија ги раздвојува елементите, а потоа со дилатација ги враќа во првобитната состојба користејќи исти елемент за структурирање. Оваа морфолошка операција вообичаено се користи за отстранување на непотребни пиксели т.е шум.

слика 8 Применување на морфолошка операција opening - отстранет е шумот (дел од јадрото во горниот десен агол), одалеченоста помеѓу јадрата е појасна, но тие се речиси во првобитна големина

Opening е идемпотентна операција, што значи дека без разлика колку итерации се направени резултатот од операцијата е ист

Closing

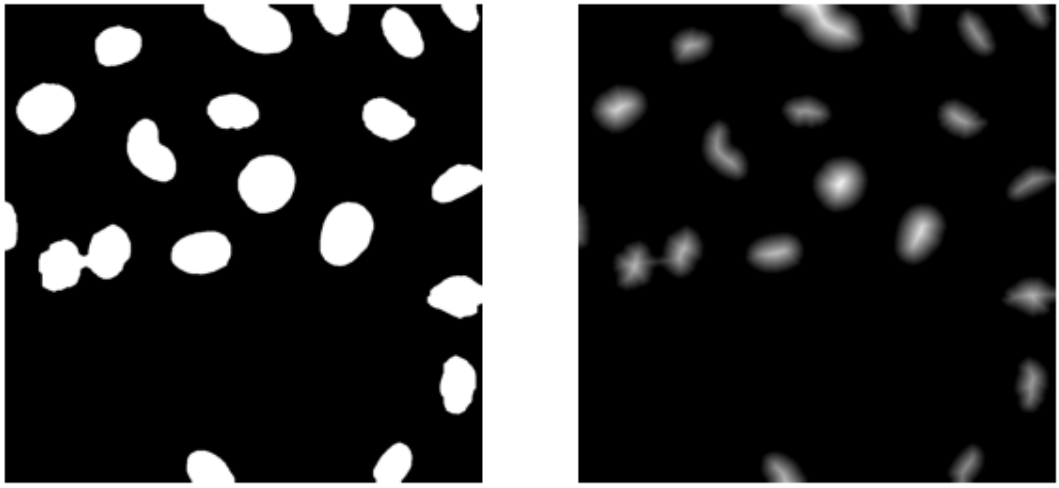
Closing е морфолошка операција која е спротивното од closing - комбинација од дилатација проследена со ерозија. Името closing или затварање доаѓа од тоа што ги пополнува празнините во елементите без притоа да им ја промени големината.

слика 9 Применување на морфолошка операција closing – пополнети се неправилностите во обликот на јадрото, а притоа јадрата ја имаат зачувано првобитната големина

Closing е идемпотентна операција, што значи дека без разлика колку итерации се направени резултатот од операцијата е ист

Пресметување на растојанија

Како што е спомнато погоре watershed е алгоритам кој ги сегментира сликите така што оригиналната слика ја трансформира во тополошка мапа. За да ја постигнеме трансформацијата ја користевме функцијата distanceTransform() од библиотеката OpenCV со аргументи оригиналната слика и вредност за distanceType односто тип на пресметување на растојанија - cv2.DIST_L2. Функцијата distanceTransform() пресметува просечна или точна далечина од секој пиксел во бинарна слика до најблискиот пиксел со вредност 0 или поточно пресметува одалеченост на секој пиксел што го формира елементот до позадината. На ваков начин пикселите поблиску до границата доиваат помала вредност од оние кои се наоѓаат во центарот на елементот имитирајќи тополошка мапа. Параметарот distanceType укажува на метриката за мерење на растојанието што во овој случај се изедначува со евклидово. Дополнително постојат повеќе различни растојанија како минковски или пак дефинирано од самиот корисник.



слика 10 Трансформирање на пример сликата во тополошка мапа

Претпроцесирање

Сликите се вчитуваат со функцијата imread() од библиотеката OpenCV во која го наведуваме патот каде се наоѓа таа слика и начинот на вчитување т.е колку канали ќе има сликата. Бидејќи станува збор за црно-бела слика ја вчитуваме како слика со еден канал која има вредности од 0 (црно) до 255(бело) и тоа го специфицираме со cv2.IMREAD_GRAYSCALE.

Секоја од сликите мора да биде претпроцесирана пред да се употреби алгоритамот. Поради тоа што работиме со операции на ниво на пиксел со цел алгоритамот да дава добри резултати потребно е сите слики да имаат исти доволно големи димензии. Слика со повеќе број на пиксели ни дава повеќе простор да извршуваме морфолошки операции. Ја избравме големината 1024x1024 и истото го направивме со функцијата resize од бибилиотеката OpenCV каде како аргументи ги проследуваме сликата и големината во облик на tuple – (1024,1024)

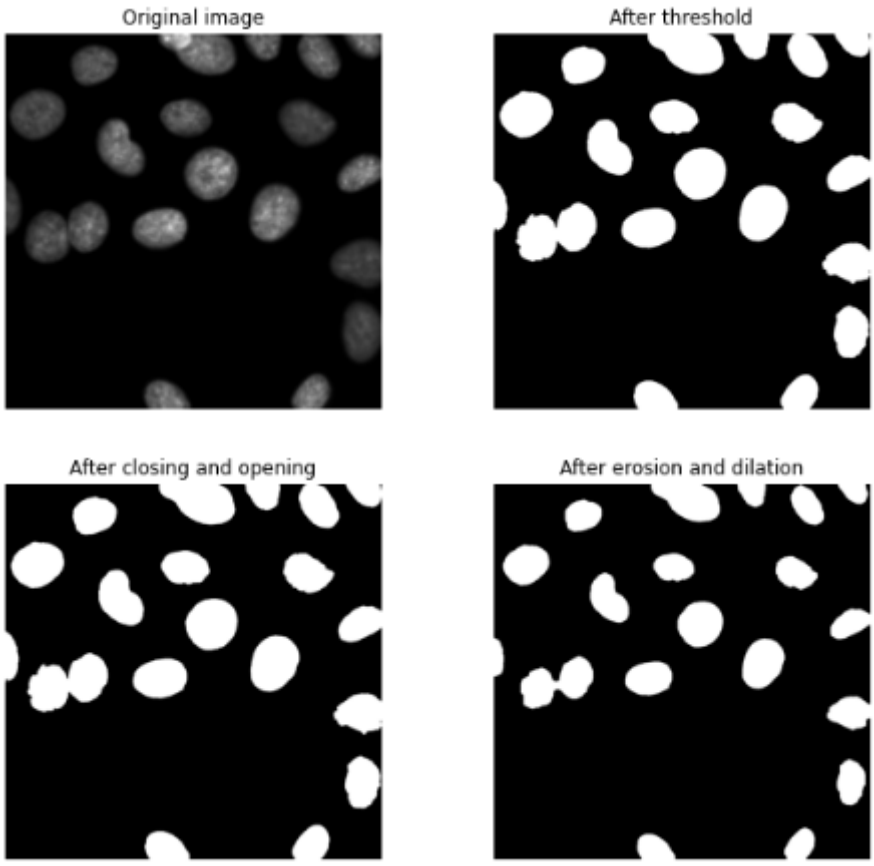
Во следниот чекор ја бинаризираме сликата – позадината во црно (вредност 0), елементите бели (вредност 1). За таа намена ја користиме функцијата thresholded и како аргумент ја

проследуваме сликата. Во функцијата ја повикуваме threshold функцијата од библиотеката OpenCV и како критериум за бинаризирање на сликата го поставуваме cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU. Сите вредности поголеми од критериумот добиваат вредност 255, а оние под критериумот вредност 0.

За да го отстраниме шумот на сликата ја користиме морфолошката операција opening и потоа морфолошката операција closing за да ги пополниме празнините. За елемент за структурирање користиме елипса со големина 3x3

По преглед на сликите можеме да заклучиме дека постојат многу јадра кои се допираат па со цел да ја намалиме нивната големина на сметка на позадината дополнително правиме 10 итерации на ерозија со елемент за структурирање 5x5. За да ги вратиме во речиси првобитна големина, но без допири извршуваме дилатација со истиот елемент за структурирање со помал број на итерациии = 6.

Големините на елементите за структурирање се одредени по проверка на повеќе различни вредности на дел од сликите – избрани се тие кои даваа најдобри резултати. Обликот на елементите за структурирање со во форма на елипса како и самите јадра што треба да се пребројени



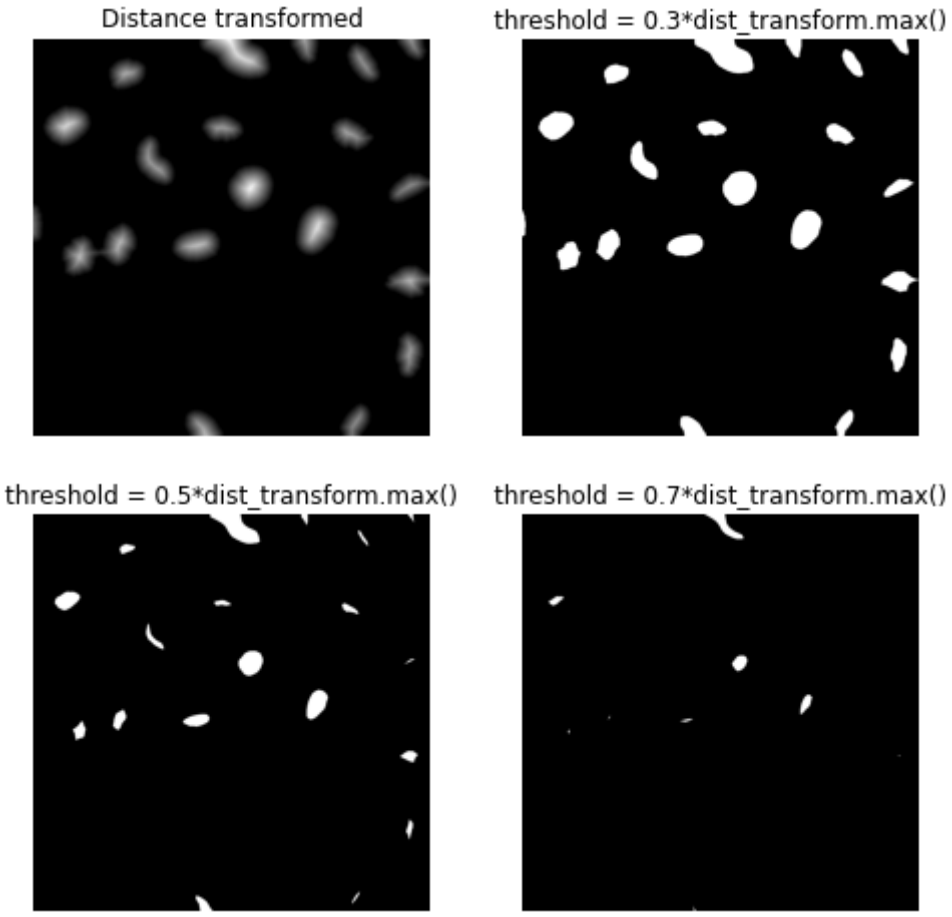
слика 11 Претпроцесирање на пример сликата

На вака претпроцесирани слики можеме да го употребиме алгоритамот.

Употреба на watershed алгоритамот за броење на јадра

По претпроцесирање на сликите потребно е да ги трансформираме во тополошки мапи за кое што ја користиме функцијата distanceTransform() за пресметување на растојанија како што е наведено погоре.

Новата бинарна слика што ја добивме се карактеризира со високи вредности во центарите на елипсовидните јадра и се пониски вредности како се наближуваат рабовите на јадрата. За нашиот проблем на пребројување на јадрата не ни е потребна границата на истите тие јадра со позадината па можеме да се сконцентрираме на повисоките вредности. Ова дополнително помага доколку некои од јадрата се допираат и покрај изведување на морфолошката операција ерозија при претпроцесирање. За да ги зачуваме повисоките вредности потребно е повторно да ги отстраниме вредносите кои не задоволуваат некој праг, т.е се помали од некоја вредности со истата функција од претпроцесирањето threshold(). Вредноста која претставува праг ја изедначуваме со 0.5*максималната вредност која ја имаат "највисоките" (најцентралните во еден елемент) пиксели. Оваа вредност е избрана по тестирање на повеќе вредности на дел од сликите, за кои оваа вредност даде најдобар резултат.

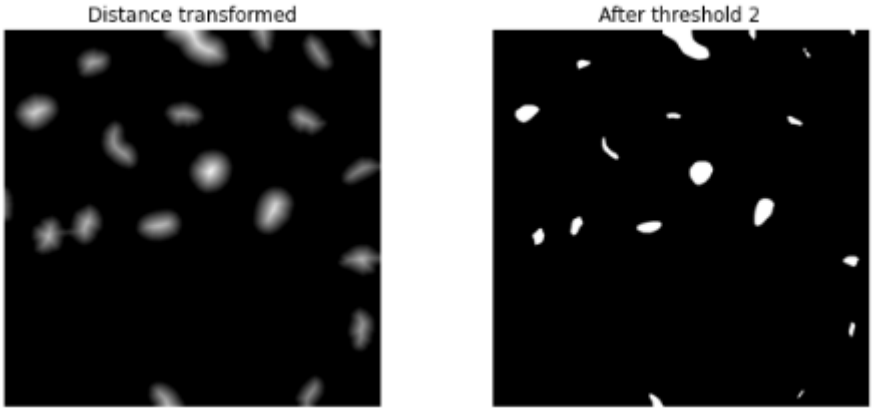


слика 12 Различни прагови во функцијата threshold

Сликата резултат од претодната функција ги содржи само планинските врвови од трансформацијата. Иако повеќе пати направивме ерозија со цел да ги одделиме јадрата кои се поклопуваат или се допираат, со цел да ги одделиме тие кои имаат многу долга заедничка граница ќе направиме уште една морфолошка операција opening. Ерозијата ќе ги оддели

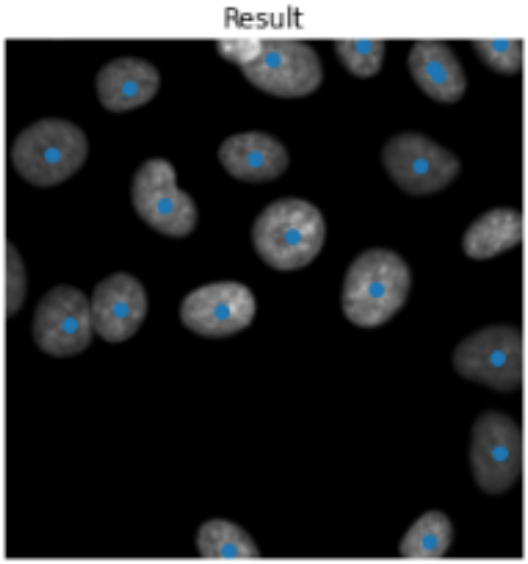
планинаските вериги во врвови, а дилатацијата ќе им ја врати речиси првобитната големина без да се допираат.

Вака добиената слика идеално би имала онолку групи на бели пиксели колку и јадра на сликата. Со цел да ги пребројиме ќе ги разгледуваме како поврзани компоненти во една слика. Ја повикуваме функцијата connectedComponentsWithStats(). Излезот на оваа функција се состои од повеќе дела, вклучувајќи лабелирана слика со различни вредности за секој од компонентите, статистики и местоположба на центроидите (центарите на секоја од компонентите). За визуелизирање на јадрата ќе ги користиме центроидите, како и за добивање на бројот на пронајдени јадра ќе ја користиме должината на листата во која се запишани центроидите.

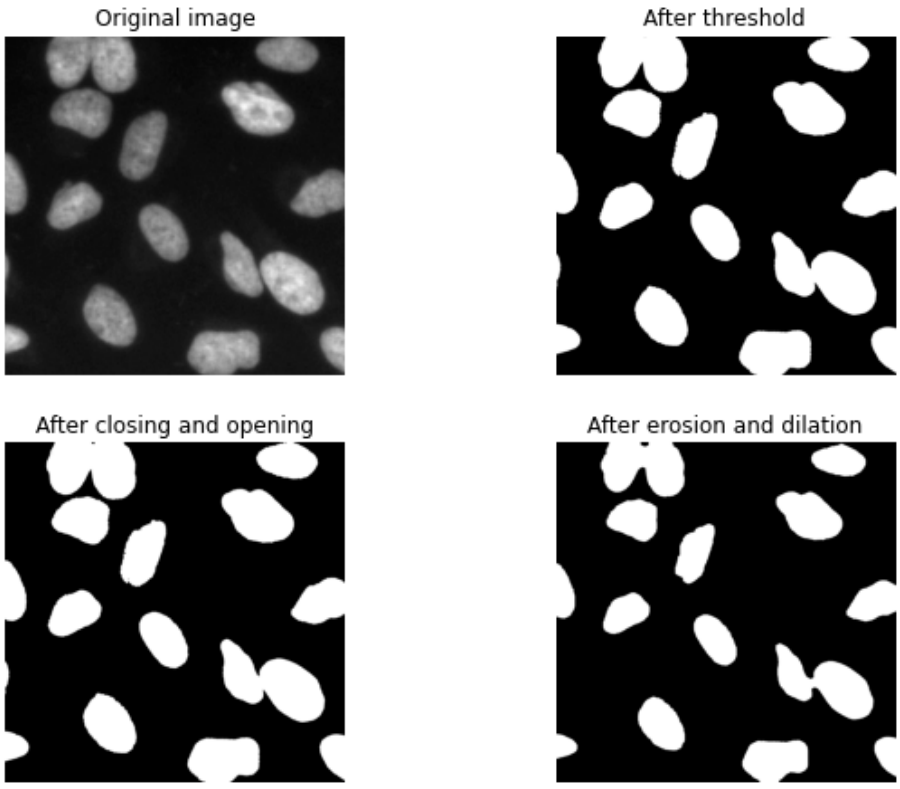


слика 13 Фазите на алгоритамот

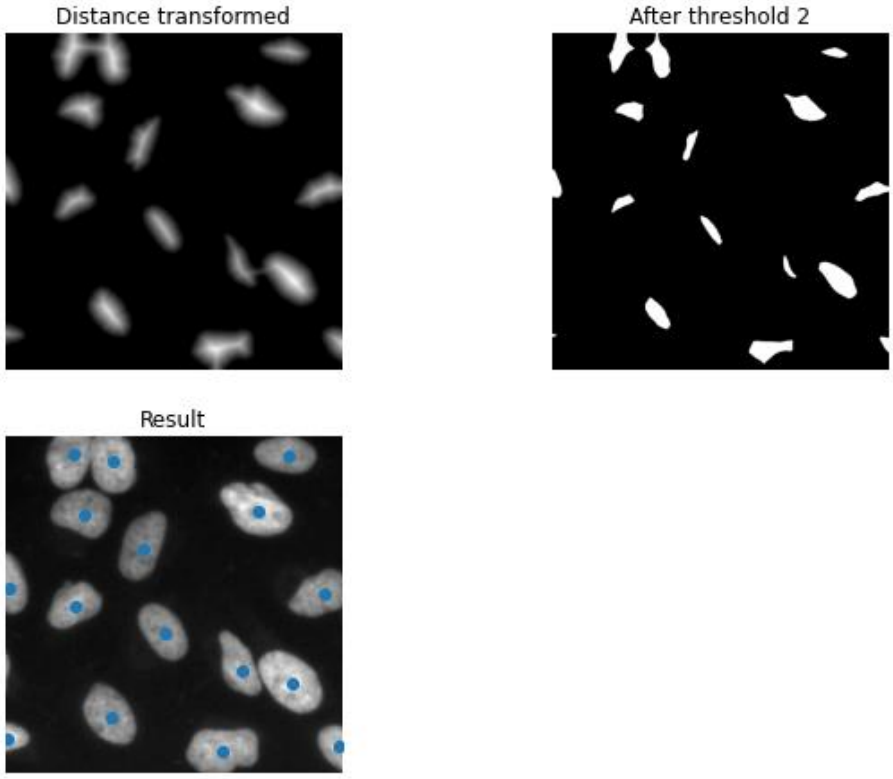
Резултати



Како што можеме да видиме од резултантната слика, а и од сите меѓуфази на алгоритамот двете јадра кои се наоѓаат во горниот дел од сликата се вбројувани како едно поради тоа што не успеавме да ги разделиме. Доколку употребевме повеќе ерозија ќе изгубевме дел од останатите клетки. Резултатот изнесува 19 јадра наспроти 20те кои се наоѓаат на сликата. Алгоритмот го пуштивме на повеќе тест слики вклучувајќи и неколку кои што имаа јадра кои се допираат/поклопуваат по со помала површина. Во продолжение се прикажени и тие резултати.



слика 14 Претпроцесирање на слика 2

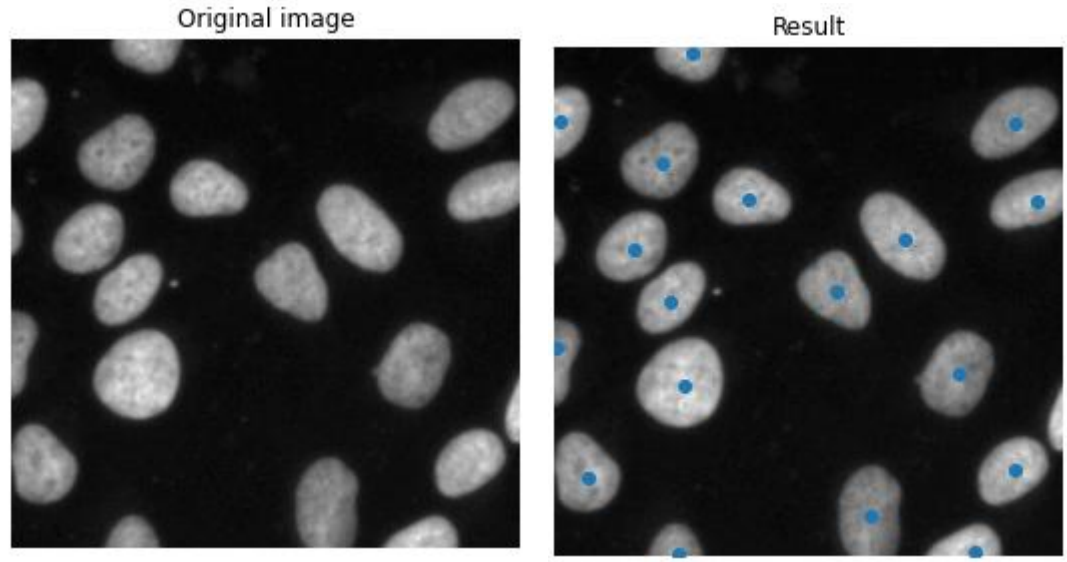


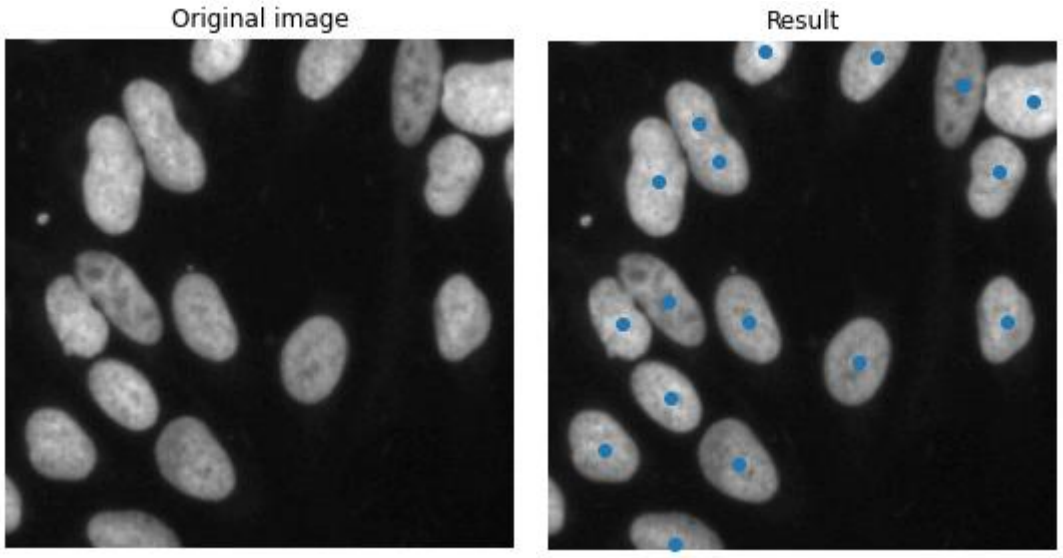
слика 15 Меѓуфази на алгоритамот и резултат на слика 2

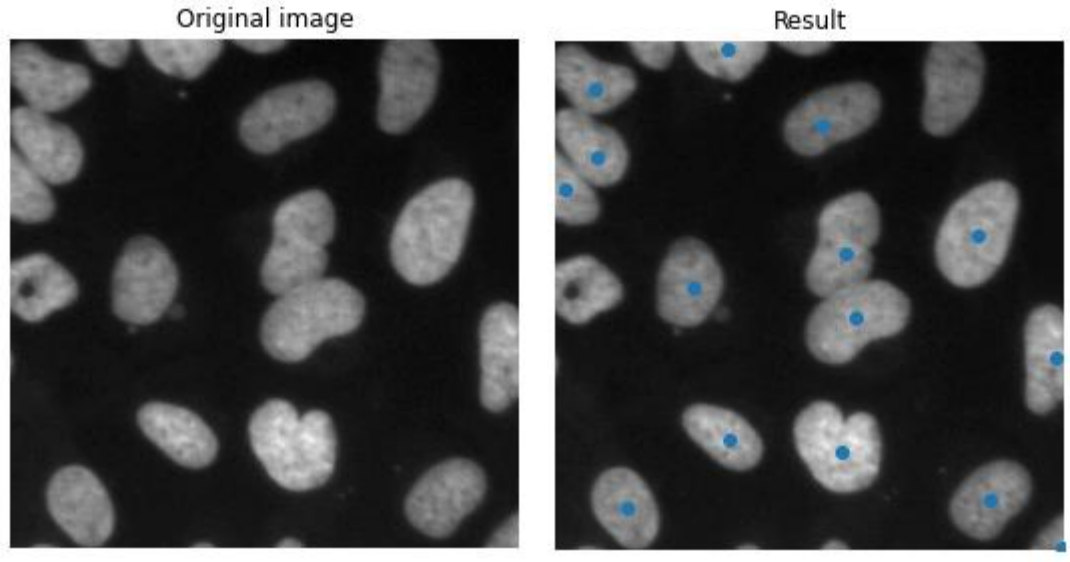
Можеме да забележиме дека доколку површината на допир повеќу две јадра е помала, алгоритмот нема никаков проблем да ги раздели овие јадра и точно да ги преброи.

Заклучок

Откако алгоритамот беше испробан на повеќе слики можеме да заклучиме дека истиот задоволително го пресметува бројот на јадра на една слика. Задоволителните перформанси се должат на константната големина на јадрата на која што можеше многу лесно да се одреди бројот и видот на морфолошки операции со цел јасно да бидат поделени јадрата. Обликот и големината на елементот на структурирање, како и изборот и бројот на итерации на морфолошките операции во склопот на претпроцесирањето се најбитниот дел од алгоритамот. Најголем проблем како и во другите алгоритми за сегментација претставува преклопувањето на јадрата, како јадрата прикажани на пример сликата. Сепак можеме да заклучиме дека дури и кај ваков проблем доколку површината на преклопување е мала, алгоритамот успешно ќе ги подели јадрата.


Original image
Result


Original image
Result


Original image
Result

Референци

[1] Featured Prediction Competition - Data Science Bowl 2018, Kaggle

[2] OpenCV Documentation

[3] Morphological Image Processing

[4] Image Segmentation with Watershed Algorithm