Факултет инжењерских наука Универзитета у Крагујевцу

Немања Д. Поповић

Сегментација монохроматских слика оптималним глобалним прагом

дипломски рад

Крагујевац, 2020.

Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Рачунарска техника и софтверско инжењерство
Ниво студија: Основне академске
Предмет: Дигитална обрада слике
Број индекса: 563/2016

Немања Д. Поповић

Сегментација монохроматских слика оптималним глобалним прагом

дипломски рад

Комисија за преглед и одбрану:
1. **др Маријана Гаврлиловић Божовић - ментор**
2._____
3._____

Датум одбране:_____
Оцена:_____

Садржај

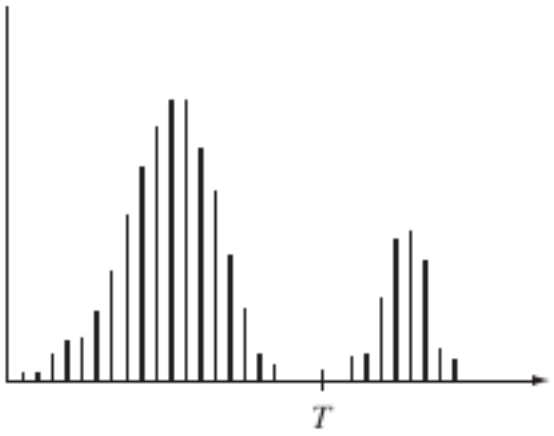
1 Увод

Термин сегментација слике се односи на групу поступака за поделу слике на регионе са сличним атрибутима. Сегментација се може применити како на слике у боји, тако и на монохроматске слике. Сегментација слике дели дату слику на објекте и позадину. Примена сегментације се завршава онда када се у апликацији детектује објекат или регион од значаја. Када се говори о обради слика, сегментација слика које садрже доста шума или сметњи представља компликован задатак који је потребно решити. Прецизност сегментације је једна веома битна ставка која такође варира од проблема који је постављен, због тога што је у неким примерима потребно само детектовати неки облик, ивицу, а у неким ситуацијама је потребно детаљно одредити детаље који су од значаја за даље процесирање података. Алгоритми за сегментацију монохроматских слика су генерално базирани на две категорије које у обзир узимају интензитет пиксела, а те категорије се односе на разлике и сличности међу пикселима. У прву групу спадају алгоритми код којих се сматра да су границе региона веома различите, што за последицу има изражене разлике у интензитету. Један од алгоритама који се базира на том принципу, јесте примена сегментације са оптималним прагом. Код друге се примењује сегментација заснована на региону. Код сегментације се могу користити боја, осветљеност, обележја попут ивице, текстуре. Сегментација има примену у обради слике, али и у компјутерској визији. Поред примене у многим областима, сегментација се примењује и у медицини, где се користи за обраду разних снимака(CT, MRI). Применом сегментације у овој области омогућава се откривање многих болести, уочавање образаца који могу послужити за даљу обраду, али и за конструисање 3Д модела. У наставку рада биће детаљно приказана сегментација применом оптималног прага коришћењем *Otsu* методе, *global treshold*. Разматрана је могућност примене реализованих алгоритама сегментације на сликама које садрже шум, као и детекција ивица коришћењем *Canny* алгоритма. Поред детаљног приказа реализације сваког алгоритма, биће приказани и резултати добијени након њихове примене на одговарајућем скупу тестних слика.

2 Алгоритми сегментације

2.1 Глобал threshod

Global threshod је алгоритам који се користи када је интензитет пиксела између објекта и позадине груписан у две доминантне групе (слика 1), али могуће је применити и *global threshod* на сликама које имају више групација.



Слика 1. Хистограм слике за примену treshold-а са једним прагом[1]

Хистограм слике даје информације о расподели нивоа сивог у слици. Сваком нивоу се може придружити број пиксела чији интензитет одговара том нивоу. Монохроматске слике које ће се разматрати у раду су представљене са 8 бита, тако да је опсег интензитета између 0 и 255. На хоризонталној оси се налазе нивои док се на вертикалној оси налази број пиксела који договара датом нивоу. У већини случајева се приступа нормализовању хистограма, што се постиже тако што се свака компонента која предтавља број пиксела са одређеним интензитетом подели са укупним бројем пиксела и на тај начин се одређује вероватноћа за сваки ниво у слици, при чему је сума свих тих вероватноћа једнака 1. На основу хистограма се могу оучити одређене особине посматране слике. Ако је хистограм слике збијен по хоризонталној оси и позициониран на левој страни графика, то значи да је слика која се анализира тамна слика. А ако је исти случај само са десне стране, то значи да је слика светла. Када се говори о распрострањености вредности на графику, ако су вредности збијене то значи да слика има слаб контраст, а ако су вредности распоређене дуж целе хоризонталне осе, онда слика има добар контраст. Начин на који се може доћи до раздвајања пиксела јесте одабир пиксела који припадају позадини и пиксели који припадају објекту. Једначина (1) која дефинише начин реализације дата је у наставку.

$$g(x,y) = \begin{cases} 1, & \text{if } f(x,y) > T \\ 0, & \text{if } f(x,y) \leq T \end{cases} \tag{1}$$

Вредност T представља *threshold*, односно ону вредност која ће помоћи при раздвајању објекта од позадине. Поступак је следећи: сегментисана слика *g(x,y)* се добија тако што се за сваку тачку *(x,y)* слике узима вредност, односно интензитет тог пиксела, *f(x,y)*, и упоређује са вредношћу прага T; ако је интензитет тог пиксела већи од T онда се у сегментисаној слици на тој позицији поставља вредност 1 и та тачка се назива *тачка објекта*, а у супротном се на тој позицији поставља вредност 0 и та тачка се назива *тачка позадине*. Ако вредност treshold-а T остаје иста током проласка кроз све тачке, онда се овакав начин реализације помоћу једначине (1) назива *global threshod*. Поред оваквог начина реализације постоје и други, када се вредност threshold-а T мења током проласка кроз слику и тада се тај начин назива *промењљив threshold*, затим када вредност threshold-а T зависи од локалних, односно суседних, пиксела такав начин реализације је познат под називом *локални threshold*. Сложенији вид реализације јесте када је потребно реализовати *threshold* помоћу

више прагова, као што је приказано једначином (2).

$$g(x,y) = \begin{cases} a, & \text{if } f(x,y) > T_2 \\ b, & \text{if } T_1 < f(x,y) \leq T_2 \\ c, & \text{if } f(x,y) \leq T_1 \end{cases} \tag{2}$$

Поступак реализације је сличан поступку као код примене *threshold*-а са једним прагом, само што се у овом случају дефинишу додатни прагови.
Агоритам са једним прагом се примењује онда када постоји уочљива граница између објекта и позадине. Ако граница није уочљива, алгоритам сегментације оптималним прагом није најпогоднији за примену. Процес одређивања глобалног прага јесте постављање почетне вредности за праг T, најчешће почетна вредност једнака је средњој вредности пиксела у целој слици, затим је потребно формирати две групе пиксела, једна која садржи пикселе чији је интензитет већи од T, а друга код које су пиксели чија је вредност мања или једнака T. Након тога, потребно је израчунати средњу вредност интензитета пикслела у обе групе. Када је израчуната средња вредност интензитета у те две групе, рачуна се нова вредност за T, тако што се саберу вредности средњег интензитета датих група и подели са два. Тај процес се понавља све док разлика између нове и претходне вредности није мања од задатог броја (број се одређује експерименталним путем).

2.2 Одређивање оптималног глобалног прага

Otsu алгоритам се базира на аутоматском рачунању глобалне вредности за праг. Циљ алгоритма јесте да врати вредност која ће омогућити поделу слике на две класе, односно на објекат и позадину. Поступак реализације се састоји од следећих корака:

1. Процесирање улазне слике (конверзија слике у ниво сивог)
2. Рачунање хистограма слике
3. Одређивање вредности за treshold T
4. Замена вредности пиксела у бело када је вредност већа од T, супротно у црно када је вредност мања од T

Идеја која се користи при реализацији јесте процесирање хистограма, односно минимизација варијансе у обе класе. Хистограм код оваквог начина реализације углавном садржи два пика који представљају слике са приметно различитим интензитетом. Једначина (3) на којој се базира алгоритам је приказана у наставку. Променљива $\sigma_w^2(t)$ представља варијансу, која се рачуна између класа које су формиране коришћењем вредности за праг, где ће се на крају тражити минимална вредност како би се пронашао оптимални праг.

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \tag{3}$$

Код једначине (3), променљива t представља праг, а вредност коју узима је из опсега од 0 до 255, (вредности које одговарају броју нивоа сивог у монохроматској слици која је предтсављена са 8 бита), коефицијенти ω_0 и ω_1 јесу вероватноће те две класе које су добијене поделом, а σ_0^2 и σ_1^2 јесу варијансе те две класе. Променљиве $\sigma_0^2(t)$ и $\sigma_1^2(t)$ се рачунају на следећи начин:

$$\sigma_0^2(t) = \sum_{i=0}^{t-1}[i - \mu_0(t)]^2 \frac{P(i)}{\omega_0(t)} \tag{4}$$

$$\sigma_1^2(t) = \sum_{i=t}^{L-1}[i - \mu_1(t)]^2 \frac{P(i)}{\omega_1(t)} \tag{5}$$

Варијанса и стандардна девијација дају информације о промени интензитета у слици, једначина за рачунање варијансе је дата једначином (4) и она гласи:

$$\sigma^2 = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}(L[m,n] - L_{sr})^2}{MN} \tag{6}$$

Променљива L_{sr}, једначина (7), представља средњу вредност интензитета слике, где је $L[m,n]$ интензитет нивоа сивог на позицији (m,n), а M,N су димензије слике.

$$L_{sr} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}L[m,n]}{MN} \tag{7}$$

Коефицијенти ω_0 и ω_1 се рачунају на следећи начин:

$$\omega_0(t) = \sum_{i=0}^{t-1} P(i) \tag{8}$$

$$\omega_1(t) = \sum_{i=t}^{L-1} P(i) \tag{9}$$

Променљива L представља број нивоа интензитета сивог, којима је дефинисана слика, пошто се ради о нивоима сивог монохроматских слика које су представљене са 8 бита, вредност који узима L је 256. *P(i)* је вероватноћа сваког пиксела да има вредност i рачуната у обе класе C_0 и C_1. Укупан

број пиксела у слици ће бити означен са n, тако да вероватноћа нивоа рачуна применом једначине (10).

$$P(i) = \frac{n_i}{n} \tag{10}$$

Променљива n_i представља број пиксела са одређеном вредношћу нивоа сивог. Интензитети пиксела који се налазе у C_0 се налазе у опсегу $[0, t-1]$, а опсег за C_1 је $[t, L-1]$.
Поред једначине (3) постоји и друга метода помоћу које се може израчунати оптимални праг, која за разлику од прве методе користи максимизацију унутар класе, а за примену ове методе је коришћена једначина (11). Разлика је та што се код једначине (3) рачунају вредности варијансе између класа, док се у једначини (11) рачуна варијанса унутар класе. У случају да је изабрана погрешна вредност за праг, варијанса неке класе би била већа што би за последицу имало погрешно одређивање *threshold*-а.

$$\begin{aligned} \sigma_b^2(t) &= \sigma^2 - \sigma_w^2(t) \\ &= \omega_0(t)(\mu_0 - \mu_T)^2 + \omega_1(t)(\mu_1 - \mu_T)^2 \\ &= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2 \end{aligned} \tag{11}$$

Променљива $\sigma_b^2(t)$ представља варијансу између класа, променљива σ^2 је укупна варијанса, променљива $\sigma_w^2(t)$ варијансу унутар класе. Вредности $\mu_0(t)$ и $\mu_1(t)$ се рачунају на следећи начин:

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} iP(i)}{\omega_0(t)} \tag{12}$$

$$\mu_1(t) = \frac{\sum_{i=t}^{L-1} iP(i)}{\omega_1(t)} \tag{13}$$

А, променљива μ_T (укупна вредност интензитетка на целој слици):

$$\mu_{(T)} = \sum_{i=0}^{L-1} iP(i) \tag{14}$$

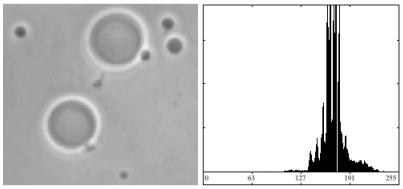
при чему важи релација:

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T \tag{15}$$

Рад ће се базирати на једначини (11). Све ове једначине служе као метод како би се дошло до глобалног прага. Након добијања вредности за глобални праг T, следећи корак јесте примена релације:

$$g(x,y) = \begin{cases} 1, & \text{if } f(x,y) > T \\ 0, & \text{if } f(x,y) \leq T \end{cases} \tag{16}$$

Након тог корака добија се бинаризована слика која је подељена на објекат и позадину. Ако је слика на којој се примењује овај метод оштећена шумом или та слика нема јасну границу која раздваја позадину и објекат, тада метода може дати погрешне резултате. Један такав пример, где нема јасне разлике је приказан на слици број 2.



Слика 2. Хистограм слике код које не постоји јасна разлика између објекта и позадине[2]

Када се примени *Otsu* метода резултат који је добијен је приказан на слици број 3,

Слика 3. Слика након сегментације оптималним глобалним прагом[3]

а након примене методе која рачуна глобални праг преко средње вредности интензитета пиксела добија се резултат приказан на слици број 4.

Слика 4. Слика након примене глобалног прага [4]

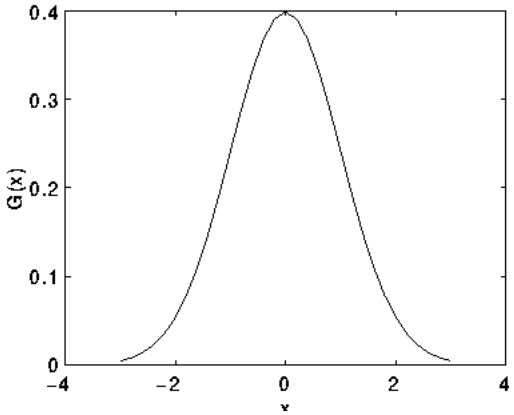
Сваки од ових случаја ће бити примењен и показан резултат на основу апликације која је реализована у оквиру ове теме.

2.3 Филтрирање слика пре примене сегментације

Сегментација слика у већини случајева захтева пре и пост процесирање, поготово пре процесирање пошто у највећем броју случајева је присуство шума неминовно. Као што је напоменуто у претходним поглављима примена метода за сегментацију зависи од стања слике која је доведена на улаз, тако да методе не дају баш добре резултате када се на улазу нађе слика са шумом. Због таквих ситуација је потребно применити пре процесирање података како би се поступци који ће се примењивати у наставку имали задовољавајуће резултате. Конкретно за филтрирање шума у овом раду ће се користити *Gaussian* филтар, а након филтрирања ће се применити нека од метода за сегментацију. *Gaussian smoothing* као резултат даје слику код које шум смањен односно филтриран, али такође доводи и до губитка детаља. Слика која се добија јесте замућена слика. Гаусова дистрибуција у 1-Д је описана једначином (17).

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{(-\frac{x^2}{2\sigma^2})} \tag{17}$$

Променљива σ представља стандардну девијацију, график дате функције је приказан на слици број 5.



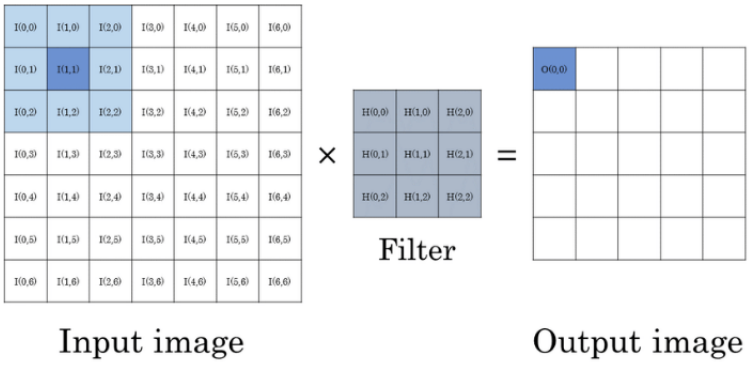
Слика 5. 1-Д Gaussian дистрибуција са

$$\sigma = 1^{[5]}$$

Када се ради о 2-Д, *Gaussian* има форму приказану једначином (18).

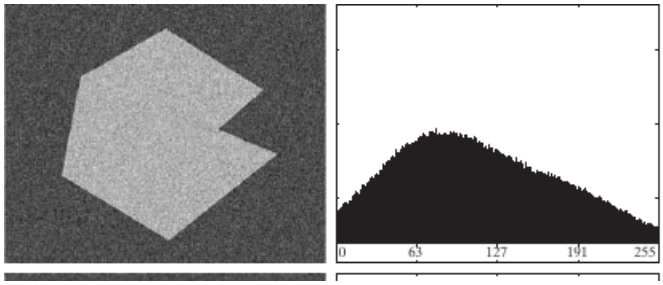
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{18}$$

Променљива која одређује степен замућења је σ. Како слика представља скуп дискретизованих вредности потребно је извршити дискретизацију *Gussian* функције како би се применила на слику. То се постиже тако што се дефинише "прозор" који садржи коефицијенте, након чега се тај "прозор" примењује на слику, тако што се пролази кроз све пикселе при чему се изводе рачунске операције и на место централног пиксела се уписује резултат. Тај поступак се спроводи кроз целу слику. На крају тог процеса се добија замућена слика, након чега се примењује нека од метода за сегментацију. Замућење помоћу *Gaussian* функције представља нискофреквентно филтрирање. У зависности од шума који се налази на слици није могуће применити сваку врсту филтра на сваку врсту шума. Процес филтрирања се примењује тако што се дефинише матрица димензија 3x3, 5x5, 7x7..., затим се итерира кроз матрицу улазне слике и рачуна се вредност која ће представљати пиксел у филтрираној слици. Поступак се може видети на слици број 6.



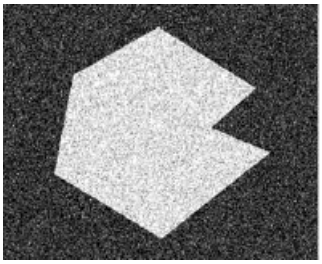
Слика 6. Процес филтрирања[6]

Тај поступак се понавља док се не дође до последњег пиксела. Примена ове методе има широку употребу, а најчешће је коришћена код детектовања ивица. Пример слике која садржи шум и њен хистограм је приказан на слици број 7.



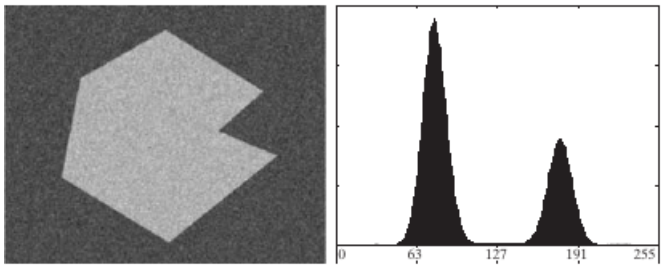
Слика 7. Слика са шумом[7]

Након примене *Otsu* методе, резултат је:



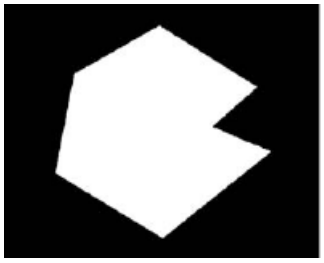
Слика 8. Otsu метода примењена на слику са шумом[8]

Када се примени Gauss филтар слика која се добија и хистограм су:



Слика 9. Изглед слике и хистограма након филтрирања[9]

А, када се примени *Otsu* метода тада је резултат:



Слика 10. Otsu на филтрираној слици[10]

Уочава се да је видљиво побољшање квалитета сегментације када се алгоритам за сегментацију примени на претходно процесирану (филтрирану) слику.

2.4 Canny алгоритам

Canny алгоритам је агоритам који се користи за детекцију ивица на слици. Поред филтрирања, одређивање ивица је још један метод у препроцесирању, који може да поправи квалитет сегментације слике. У оквиру секције број 4 биће дат приказ слике на којој је примењен *Canny* алгоритам пре него што је на дату слику примењена нека од метода за сегментацију са оптималним прагом, као и након примене неке од тих метода. Дати алгоритам се базира на три основне карактеристике:

1. *Мала стопа грешке*: Све ивице би требало да буду пронађене и да те ивице буду што ближе стварним ивицама;
2. *Тачке ивица би требало да буду тачно локализоване*: Растојање између центра ивице која је детектована и центра стварне ивице би требало да буде минимално;
3. *Јединствено означавање тачке за ивицу*: Потребно је означити само једном тачку за ивицу, како би се избегло стварање лажних ивица:

Главни циљ јесте покушај проналажења оптималног модела који ће испуњавати дате захтеве. Кораци које је потребно предузети приликом реализовања датог алгоритма су:

1. Смањење шума
2. Рачунање градијента
3. Потискивање не-максимума
4. Дупли *threshold*
5. Праћење ивица помоћу хистерезиса

1. Смањење шума
Као што је објашњено у секцији 2.3, филтрирање је битан поступак у препроцесирању слика. Слике у највећем броју случаја садрже неку врсту шума. Присуство шума доводи до појаве лажних ивица, које могу изазвати погрешно детектовање приликом примене алгоритма за детекцију ивица. Поступак који се користи у овом раду јесте објашњен у секцији 2.3. Користи се *Gauss*-ов филтар за смањење шума. Величина прозора који се користи за итерацију кроз слику може бити разних димензија (3x3, 5x5,7x7, ...). У зависности величине прозора зависи и ефекат замућивања, што је прозор мањи то је ефекат мање видљив. Једначина *Gauss*-овог филтра за прозор величине *(2k+1)*(2k+1)* је:

$$H_{ij} = \frac{1}{2\pi\sigma^2} e^{-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}}; 1 \leq i; j \leq (2k+1) \tag{19}$$

2. Рачунање градијента

Током овог корака се рачуна интензитет ивице и правац коришћењем градијента. Потребно је покрити хоризонталне, вертикалне и дијагоналне ивице у замућеној слици. Применом *Sobel*-овог оператора добија се први извод у хоризонталном (G_x) и вертикалном (G_y) правцу. Матрице које се користе за рачунање првог извода су:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

На основу овога се градијент и правац ивице може рачунати помоћу:

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{20}$$

$$\Theta = \arctan(\frac{G_y}{G_x}) \tag{21}$$

G је *магнитуда*(*дужина*) вектора ∇f . При чему ∇f представља градијент слике f. Једначина која дефинише градијент је дата под бројем (22).

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{22}$$

Правац градијентног вектора је дат једначином:

$$\alpha(x, y) = \arctan[\frac{g_y}{g_x}] \tag{23}$$

Како се обрада базира на дискретним вредностима потребно је апроксимирати једначине које имају примену у континуалном домену. Једначина која приказује апроксимацију је једначина (24),

$$G[m, n] \approx \sqrt{(f[m, n] - f[m + 1, n])^2 + (f[m, n] - f[m, n + 1])^2} \tag{24}$$

или ако се користе апсолутне вредности једначина (25).

$$G[m, n] \approx |f[m, n] - f[m + 1, n]| + |f[m, n] - f[m, n + 1]| \tag{25}$$

Као резултат овог корака добијају се ивице које су танке, али се добијају и ивице које су дебеле. То се јавља због тога што ивице у том тренутку немају једнинствену вредност, већ имају вредност која је добијена након рачунских операција. Тако да ће се следећи корак базирати на отклањању овог недостатка, тј. вршиће се поступак такав да ће ивице имати исту вредност за интензитет.

3. Потискивање не-максимума
Главни задатак овог корака јесте истањивање ивица које су добијене кораком 2. Алгоритам се базира на томе да се прође кроз све пикселе у градијентној матрици и да се пронађу пиксели са вредностима које одступају од вредности пиксела са карактеристикама као тај пиксел. Тако да су кораци:

1. Упоредити вредност интензитета тренутног пиксела са вредношћу интензитета пиксела у позитивном и негативном правцу градијента.

2. Ако је интензитет тренутно посматраног пиксела већи од вредности пиксела у истом правцу (рецимо орјентисан је у правцу вертикалне осе, тако да ће се упоређивати са пикселом изнад и испод себе), онда ће се вредност тренутног пиксела променити, у супротном ће остати иста.

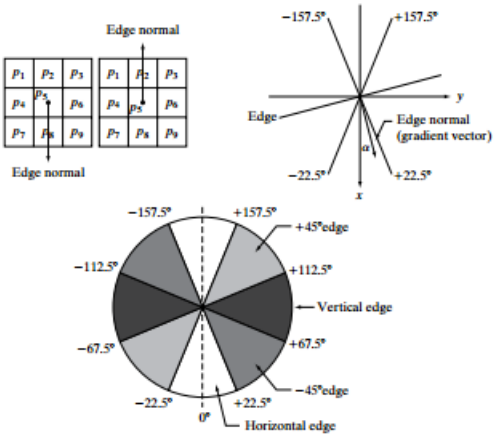
Такође, додатни начин јесте упоређивање вредности угла градијента у односу на суседне пикселе, графички приказ је дат на слици број 11:

1. Ако је заокругљена вредност градијентног угла 0° (правац је север-југ), тачка ће бити разматрана као вредност ивице, ако је вредност магнитуде градијента тог пиксела већа од вредности магнитуде градијента пиксела који је у правцу исток-запад.

2. Ако је заокругљена вредност градијентног угла 90° (правац је исток-запад), тачка ће бити разматрана као вредност ивице, ако је вредност магнитуде градије- нта тог пиксела већа од вредности магнитуде градијента пиксела који је у правцу север-југ.

3. Ако је заокругљена вредност градијентног угла 135° (правац је североисток-југозапад), тачка ће бити разматрана као вредност ивице, ако је вредност магни- туде градијента тог пиксела већа од вредности магнитуде градијента пиксела који је у правцу северозапад-југоисток.

4. Ако је заокругљена вредност градијентног угла 45° (правац је северотзапад-југоисток), тачка ће бити разматрана као вредност ивице, ако је вредност магнитуде градијента тог пиксела већа од вредности магнитуде градијента пиксела који је у правцу североисток-југозапад.



Слика 11. Процес одређивања ивица[11]

Резултат претходних корака јесу тање ивице, али се може приметити варијација у интензитету неких ивица, тако да се наредна два корака примењују како би се превазишао тај проблем.

4. Дупли threshold

Применом дуплог *threshold*-а се детектују три врсте пиксела:

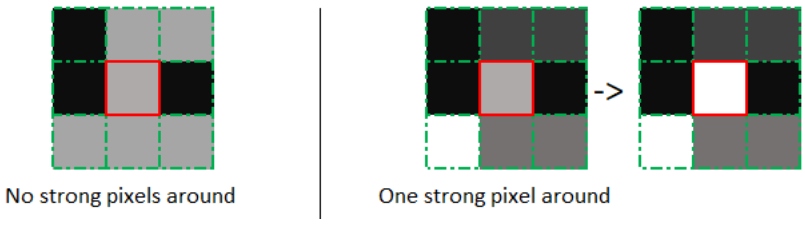
- Јаки - пиксели који имају висок интензитет и који ће се сигурно наћи у крајњој мапи ивица;
- Слаби - пиксели чији интензитет није толико висок да се смратра јаким, пак ни толико низак да се сматра не-релевантним;
- Не-релевантни - они који сигурно не представљају ивицу

Тако да дупли *threshold* представља висок и низак *threshold*.

- Висок детектује пикселе који су сигурно ивични;
- Низак детектује пискеле који нису ивични;
- Сви остали пиксели који се налазе између вредности горњег и доњег прага ће бити разматрани тако да се одреди да ли припадају ивичним пикселима или не.

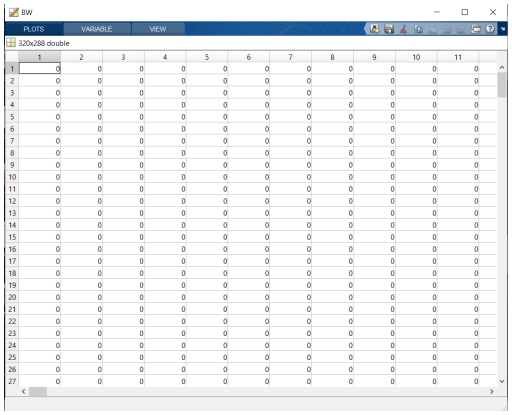
5. Праћење ивица помоћу хистерезиса

Поступак се састоји од провере да ли дати пиксел припада ивици или не припада (слика 12). То се постиже тако што се проверавају суседни пиксели, ако је бар један од суседних пиксела ивични дати пиксел се проглашава ивичним.



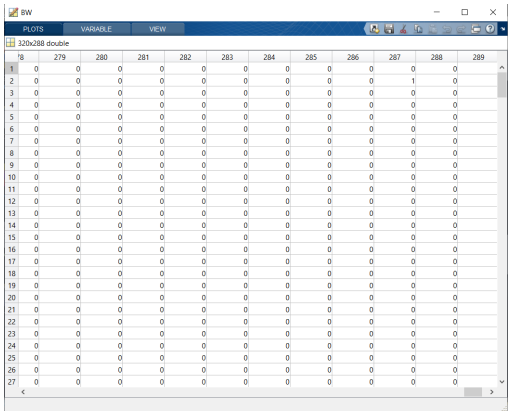
Слика 12. Основа за поделу пиксела[12]

Реализација овог корака је приказана у секцији 3.4, а испод је приказан само делимичан изглед матрица ивичних пиксела током овог процеса. Матрице које су приказане показују како се током итерација мења њихова вредност при детектовању ивица и како у зависности од вредности се проглашавају ивичним или не. На почетку матрица садржи све нуле (слика 13),



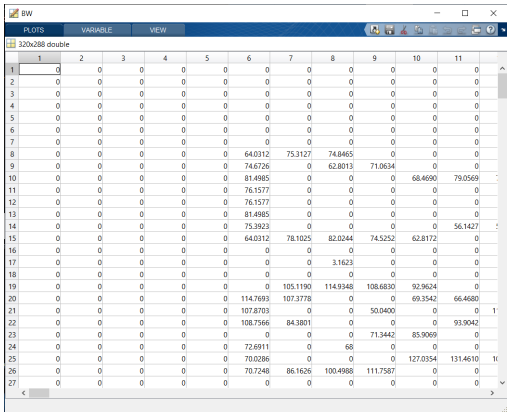
Слика 13. Матрица у коју ће се уписивати детектоване ивице

Након проласка кроз прве две итерације унутар петљи, унитар којих се проверавају ивице, садржај матрице је следећи:



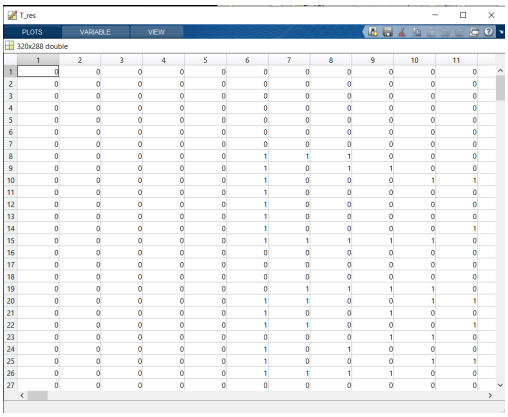
Слика 14. Детекција ивице

Након проласка кроз целу матрицу резултат је:



Слика 15. Матрица са свим потенцијалним ивичним пикселима

Када се провери услов за препознавање пиксела(слаби, јаки, без утицаја) резултат је:



Слика 16. Матрица која садржи само праве ивичне пикселе

3 Начин реализације

За практичну реализацију пројкета је коришћен програмски језик *MatLab* и развојно окружење *MatLab*-а, при чему је верзија која је коришћена *MatLab2017a*. У наставку ће бити објашњена реализација сваке скрипте посебно независно од *GUI* апликације. У оквиру *GUI* апликације ће бити објашњен начин реализације и дати приказ изгледа исте.

3.1 Глобал threshold

Алгоритам на којем је базирана релизација дате методе је:

Algorithm 1: Глобал *threshold*

улаз : Слика на којој ће се применити дата метода
излаз: Бинаризована слика на којој је примењена дата метода

1 Постави почетну вредност за treshold T;
2 Примени сегментацију слике коришћењем T у једначини (1). Резултат ће бити две групе пиксела : G_1 која садржи пикселе чији је интензитет > T, и G_2 која садржи пикселе чије је интензитет ≤ T ;
3 Ирачунај средњу вредност интензитета пиксела m_1 и m_2 за пикселе из G_1 и G_2 респективно ;
4 Израчунај нову вредност за *threshold* T

$$T = \frac{1}{2}(m_1 + m_2);$$

Понављај кораке од 2 до 4 све док је разлика између вредности T у суксецивним итерацијама мања од предефинисане вредности ΔT;

Препоручује се да почетна вредност за *threshold* , T, буде средња вредност интензитета пиксела у целој слици.
Код који се базира на овом алгоритму је дат у наставку.
Први корак у имплементацији јесте учитавање слике, затим конверзија дате слике у слику која садржи само нивое сивог. Затим се приказује учитана слика која је конвертована и њен хистограм. Потребно је дефинисати промењиве које ће се користити при реализацији.

```
I = imread('IMG-0001-00001.jpg');
I = rgb2gray(I);
I2 = I;
figure;
imshow(I);
set(gcf, 'Name', 'Ulazna slika');
figure;
imhist(I);
set(gcf, 'Name', 'Histogram ulazne slike');
T = sum(I2(:))/(size(I2,1) * size(I2,2));
T0 = 0;
m1 = 0;
m2 = 0;
```

Код. Учитавање и сетовање почетних вредности

Може се уочити да је за почетну вредност прага узета средња вредност интензитета пиксела са слике у примеру. Овим поступком је дефинисан први корак из алгоритма. Након овог корака се прелази на релизацију корака од 2-5.

```
while(T-T0 > 10^-5)
    T0 = T;
    g1 = I2 > T;
    g2 = I2 <= T;
    m1 = sum(I2(g1)) / sum(g1(:) == 1);
    m2 = sum(I2(g2)) / sum(g2(:) == 1);
    T = (m1+m2)/2;
end
```

Код. Реализација корака 2-5 алгоритма 1

Променљиве g_1 и g_2 не садрже директно вредности пиксела који су мање или веће од вредности прага, већ садрже 0 или 1, где представљају индексе код којих је испуњен услов. Након тога се прелази на рачунање средње вредности пиксела распоређених у две групе и након тога се рачуна нова вредност за *threshold*. Број који одређује останак у петљи је 10^{-5}. Ова вредност се утврђује експерименталним путем и представља променљиву ΔT односно разлику која је услов за заустављање овог процеса. Када је израчуната коначна вредност за *threshold*, примењује се на улазну слику.

```
tresh = T;
for i = 1 : size(I2,1)
    for j = 1 : size(I2,2)
        if(I2(i,j) > tresh)
            I2(i,j) = 1;
        else
            I2(i,j) = 0;
        end
    end
end
I2 = logical(I2);
figure;
imshow(I2);
set(gcf, 'Name', 'Izlazna slika');
```

Код. Реализација сегментације након добијања коначне вредности за treshold

На крају је потребно извршити конверзију излазне слике пошто садржи само 0 и 1. Применом овог кода је реализован алгоритам под бројем 1.

3.2 Otsu метода

Алгоритам на којем се базира *Otsu*-ов метод је :

Algorithm 2: *Otsu* метода

улаз : Слика на којој ће се применити дата метода
излаз: Бинаризована слика на којој је примењена дата метода

1 Израчунај хистограм улазне слике и вероватноће за сваки ниво сивог;
2 Иницијализуј почетне вредности $\omega_i(0)$ и $\mu_i(0)$;
3 Проћи кроз све вредности *threshold*-а t=1...максимални интензитет

1. Промени вредности ω_i и μ_i
2. Израчунај: $\sigma_b^2(t)$

;
4 Коначна вредност за *threshold* одговара максимуму $\sigma_b^2(t)$;
5 Примени једначину (1) са коначном вредношћу *threshold*-а ;

Након учитавања слике потребно је израчунати хистограм слике и одредити вероватноће за сваки ниво. Код који реализује задате захтеве се налази у наставку.

```
I = imread('IMG-0001-00001.jpg');
I = rgb2gray(I);
I2 = I;
figure;
imshow(I);
set(gcf, 'Name', 'Ulazna slika');


hist = imhist(I2);
figure;
imhist(I);
set(gcf, 'Name', 'Treshold ulazne slike');

sizeImage = size(I2,1) * size(I2,2);

max = 0;

for i = 1:256
    P(i) = hist(i) / sizeImage;
end
```

Код. Учитавање слике, рачунање хистограма и вероватноћа

На почетку је извршена конверзија слике у слику која садржи само нивое сивог. Након тога је потребно иницијализовати промењиве из корака 2 (алгоритам 2).

```
w1 = 0;
w2 = 0;
m1 = 0;
m2 = 0;
```

Код. Иницијализација промењивих

Након иницијализације приступа се реализацији петље у којој се одвија главни део алгоритма, а то је промена вредности промењивих ω_i на основу једначина (8) и (9) респективно, као и промена μ_i на основу једначина (12) и (13) респективно,као и рачунање вредности $\sigma_b^2(t)$ на основу једначине **(11)**.

```
for i = 2:256

    w1 = sum(P(1:i));
    w2 = sum(P(i+1:256));

    mi1 = dot([0:i-1],P(1:i))/w1;
    mi2 = dot([i:255],P(i+1:256))/w2;
    sigma = w1*w2*((mi1-mi2)^2);
    if (sigma > max)
        max = sigma;
        tresh = i - 1;
    end
end
```

Код. Реализација корака 3 и 4 алгоритма 2

Након одређивања вредности за *threshold* потребно је применити једначину 1 како би се употпунила реализација датог алгоритма.

```
for i= 1 : size(I2,1)
    for j = 1 : size(I2,2)
        if (I2(i,j) > tresh)
            I2(i,j) = 1;
        else
            I2(i,j) = 0;
        end
    end
end
I2 = logical(I2);
figure;
imshow(I2);
set(gcf, 'Name', 'Izlazna slika');
```

Код. Реализација петог корака алгоритма 2

3.3 Филтрирање слика пре примене сегментације

Алгоритам у оквиру којег ће се прво применити филтрирање, па затим сегментација:

Algorithm 3: Филтрирање пре сегментације

улаз : Слика на којој ће се применити дата метода
излаз: Бинаризована слика на којој је примењена дата метода

1 Додај шум на улазну слику;
2 Иницијализуј почетне вредности;
3 Примени филтрирање ;
4 На филтрирану слику примени алгоритам 1 или алгоритам 2;

У оквиру овог алгоритма се може приметити да је први корак додавање шума, тај корак је само неопходан у овом случају када је улазана слика без шума, па како би се симулирао овакав случај потребно је додати шум. Реализација је следећа:

```
I = imread('IMG-0001-00001.jpg');
I = rgb2gray(I);
I2 = I;

figure;
imshow(I2);
set(gcf, 'Name', 'Ulazna slika');

figure;
imhist(I2);
set(gcf, 'Name', 'Histogram ulazne slike');


I3 = imnoise(I2,'Gaussian',0.04,0.003);
figure;
imshow(I3);
set(gcf, 'Name', 'Slika sa Gausovim sumom');
```

Код. Додавање шума на улазну слику

Следећи корак је иницијализација променљивих.

```
I3 = double(I3);
sigma = 1.76;

sz = 1;
[x,y]=meshgrid(-sz:sz,-sz:sz);

M = size(x,1);
N = size(y,1);

Exp_comp = -(x.^2+y.^2)/(2*sigma*sigma);
Kernel= exp(Exp_comp)/(2*pi*sigma*sigma);

Output=zeros(size(I3));
%Pad the vector with zeros
I3 = padarray(I3,[sz sz]);
```

Код. Иницијализација променљивих

Једна од главних промењљивих у коду изнад јесте промењљива *sigma*. Следећи корак јесте примена филтрирања, поступак је дефинисан у оквиру секције 2.3.

```
for i = 1:size(I3,1)-M
    for j =1:size(I3,2)-N
        Temp = I3(i:(i+M)-1,j:(j+M)-1).*Kernel;
        Output(i,j)=sum(Temp(:));
    end
end
%Image without Noise after Gaussian blur
Output = uint8(Output);
figure,imshow(Output);
```

Код. Филтрирање слике

Након филтрирања потребно је изабрати једну од метода за сегментацију, алгоритам 1 или алгоритам 2, на овом примеру је изабран алгоритам 1.

```
I2 = I3;
%%%

T = sum(I2(:))/(size(I2,1) * size(I2,2));
T0 = 0;
m1 = 0;
m2 = 0;


while(T-T0 > 10^-5)
    T0 = T;
    g1 = I2 > T;
    g2 = I2 <= T;

    m1 = sum(I2(g1)) / sum(g1(:) == 1);
    m2 = sum(I2(g2)) / sum(g2(:) == 1);

    T = (m1+m2)/2;
end
tresh = T;
for i = 1 : size(I2,1)
    for j = 1 : size(I2,2)

        if(I2(i,j) > tresh)
            I2(i,j) = 1;
        else
            I2(i,j) = 0;
        end
    end
end
I2 = logical(I2);
figure;
imshow(I2);
set(gcf, 'Name', 'Izlazna slika');
```

Код. Примена алгоритма 1 на филтрирану слику

3.4 Canny алгоритам

Алгоритам за детекцију ивица (*Canny*):

Algorithm 4: *Canny*

улаз : Слика на којој ће се применити дата метода
излаз: Слика на којој је примењена дата метода

1 Филтрирај улазну слику;
2 Израчунај градијент;
3 Потискивање не максимума;
4 Дупли *threshold*;
5 Праћење ивица помоћу хистерезиса;

Опис сваког корака из алгоритма је приказан у секцији 2.4, тако да ће у наставку бити приказан код који је коришћен при реализацији.

```
I = imread('IMG-0001-00001.jpg');
I2 = I;
figure, imshow(I2);
I2 = rgb2gray(I2);
I2 = double (I2);
%Value for Thresholding
T_Low = 0.05;
T_High = 0.12;

I3 = double(I3);
sigma = 1.76;

sz = 1;
[x,y]=meshgrid(-sz:sz,-sz:sz);

M = size(x,1);
N = size(y,1);

Exp_comp = -(x.^2+y.^2)/(2*sigma*sigma);
Kernel= exp(Exp_comp)/(2*pi*sigma*sigma);
```

Код. Иницијализација променљивих

Први корак из алгоритма 4:

```
Output=zeros(size(I3));
%Pad the vector with zeros
I3 = padarray(I3,[sz sz]);

%Convolution
for i = 1:size(I3,1)-M
    for j =1:size(I3,2)-N
        Temp = I3(i:(i+M)-1,j:(j+M)-1).*Kernel;
        Output(i,j)=sum(Temp(:));
    end
end
%Image without Noise after Gaussian blur
Output = uint8(Output);
```

Код. Филтрирање слике

Други корак алгоритма 4.

```
%Filter for horizontal and vertical direction
I_x = [-1, 0, 1; -2, 0, 2; -1, 0, 1];
I_y = [1, 2, 1; 0, 0, 0; -1, -2, -1];
%Convolution by image by horizontal and vertical filter
Filtered_X = conv2(Output, I_x, 'same');
Filtered_Y = conv2(Output, I_y, 'same');
%Calculate directions/orientations
theta = atan2 (Filtered_Y, Filtered_X);
theta = theta*180/pi;
[M,N] = size(Output);
```

Код. Рачунање градијента

Трећи корак алгоритма 4.

```
for i=1:M
    for j=1:N
        if (theta(i,j)<0)
            theta(i,j)=360+theta(i,j);
        end;
    end;
end;
theta2=zeros(M, N);
%Adjusting directions to nearest 0, 45, 90, or 135 degree
for i = 1  : M
    for j = 1 : N
        if ((theta(i, j) >= 0 ) && (theta(i, j) < 22.5)
         || (theta(i, j) >= 157.5) && (theta(i, j) < 202.5)
         || (theta(i, j) >= 337.5) && (theta(i, j) <= 360))
            theta2(i, j) = 0;
        elseif ((theta(i, j) >= 22.5) && (theta(i, j) < 67.5)
         || (theta(i, j) >= 202.5) && (theta(i, j) < 247.5))
            theta2(i, j) = 45;
        elseif ((theta(i, j) >= 67.5 && theta(i, j) < 112.5)
         || (theta(i, j) >= 247.5 && theta(i, j) < 292.5))
            theta2(i, j) = 90;
        elseif ((theta(i, j) >= 112.5 && theta(i, j) < 157.5)
         || (theta(i, j) >= 292.5 && theta(i, j) < 337.5))
            theta2(i, j) = 135;
        end;
    end;
end;
%Calculate magnitude
magnitude = (Filtered_X.^2) + (Filtered_Y.^2);
magnitude2 = sqrt(magnitude);
BW = zeros (M, N);
%Non-Maximum Supression
for i=2:M-1
    for j=2:N-1
        if (theta2(i,j)==0)
            BW(i,j) = (magnitude2(i,j) == max([magnitude2(i,j),
             magnitude2(i,j+1), magnitude2(i,j-1)]));
        elseif (theta2(i,j)==45)
```

```
            BW(i,j) = (magnitude2(i,j) == max([magnitude2(i,j),
            magnitude2(i+1,j-1), magnitude2(i-1,j+1)]));
        elseif (theta2(i,j)==90)
            BW(i,j) = (magnitude2(i,j) == max([magnitude2(i,j),
            magnitude2(i+1,j), magnitude2(i-1,j)]));
        elseif (theta2(i,j)==135)
            BW(i,j) = (magnitude2(i,j) == max([magnitude2(i,j),
            magnitude2(i+1,j+1), magnitude2(i-1,j-1)]));
        end;
    end;
end;
BW = BW.*magnitude2;
figure, imshow(BW);
```

Код. Non-maximum suppression

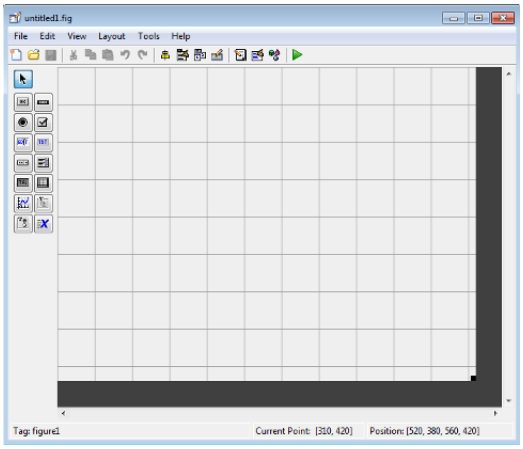
Четврти и пети корак алгоритма 4.

```
%Hysteresis Thresholding
T_Low = T_Low * max(max(BW));
T_High = T_High * max(max(BW));
T_res = zeros (M, N);
for i = 1   : M
    for j = 1 : N
        if (BW(i, j) < T_Low)
            T_res(i, j) = 0;
        elseif (BW(i, j) > T_High)
            T_res(i, j) = 1;
        %Using 8-connected components
        elseif ( BW(i+1,j)>T_High || BW(i-1,j)>T_High
          || BW(i,j+1)>T_High || BW(i,j-1)>T_High
          || BW(i-1, j-1)>T_High || BW(i-1, j+1)>T_High
          || BW(i+1, j+1)>T_High || BW(i+1, j-1)>T_High)
            T_res(i,j) = 1;
        end;
    end;
end;
I4 = uint8(T_res.*255);
%Show final edge detection result
figure, imshow(I4);
```

Код. Дупли threshold и праћење ивица помоћу хистерезиса

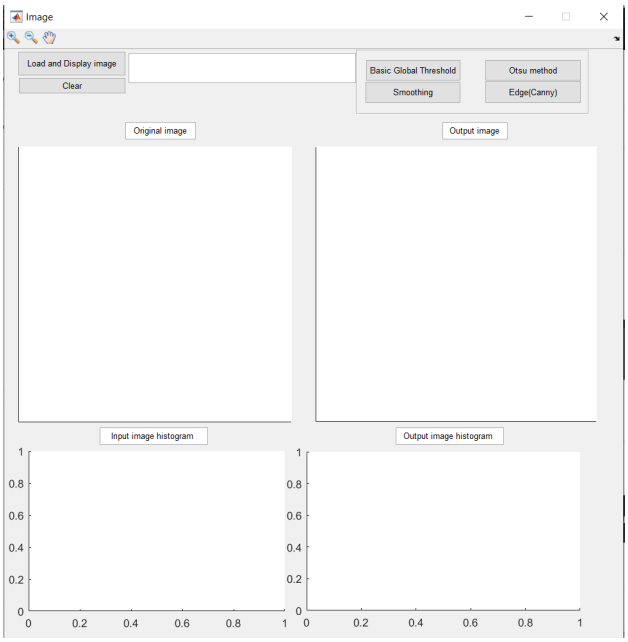
3.5 GUI апликација

За развој *GUI* апликације је коришћена библиотека *guide*. Ова библиотека омогућава избор компоненти и позиционирање у оквиру прозора (слика 17). Када је изабрана одређена компонента, код за креирање дате компоненте се аутоматски генерише у позадини, где је након тога потребно задати функционалност коју је потребно реализовати.



Слика 17. Прозор за креирање GUI апликације

Изглед апликације која је реализована за потребе овог рада је приказана на слици број 18.



Слика 18. Изглед апликације

У апликацији се уочавају шест *button*-а, четири прозора за приказ улазне слике, хистограма улазне слике, излазне слике као и хистограм излазне слике. Учитавање слике се постиже преко *button*-а *Load and Display image*, реализација је приказана на слици број 19.

```
% --- Executes on button press in load.
function load_Callback(hObject, eventdata, handles)
[rawname, rawpath] = uigetfile({'*.jpg'; '*.png'; '*.jpeg';'*.tiff'; '*.bmp'}, 'Select image');
fullname = [rawpath rawname];
set(handles.edit1, 'String', fullname);

I = imread(fullname);
I = rgb2gray(I);
axis(handles.axes1);
hold off
imshow(I,'Parent', handles.axes1);

axis(handles.axes3);
c = imhist(I);
bar(c, 'Parent', handles.axes3);
setappdata(0,'sendData',I)
```

Слика 19. Код за учитавање слике

Омогућено је учитавање слика које су формата *.jpg, .png, .jpeg, .bmp, .tif*. Следећа функционалност која је реализована јесте *Clear*. Кликом на *Clear* button се чисте сви прозори и промењиве које су у позадини сетоване. Реализација дате функције је приказана на слици број 20.

```
% --- Executes on button press in clear.
function clear_Callback(hObject, eventdata, handles)
handles.data = [];
I = 0;
arrayfun(@cla,findall(0,'type','axes'));
setappdata(0,'sendData',I);
set(handles.edit1, 'String', '');
```

Слика 20. Код за функцију clear

Реализација глобалног прага у оквиру *GUI* апликације је приказана на слици број 21.

```
% --- Executes on button press in basicTreshold.
function basicTreshold_Callback(hObject, eventdata, handles)
I = getappdata(0,'sendData');
if(I == 0)
    clear I;
    f = warndlg('Please, load image!', 'Warning');
else
    I2 = I;
    T = sum(I2(:))/(size(I2,1) * size(I2,2));
    T0 = 0;
    m1 = 0;
    m2 = 0;
    while(T-T0 > 10^-5)
        T0 = T;
        g1 = I2 > T;
        g2 = I2 <= T;
        m1 = sum(I2(g1)) / sum(g1(:) == 1);
        m2 = sum(I2(g2)) / sum(g2(:) == 1);
        T = (m1+m2)/2;
    end
    tresh = T;
    for i = 1 : size(I2,1)
        for j = 1 : size(I2,2)
            if(I2(i,j) > tresh)
                I2(i,j) = 1;
            else
                I2(i,j) = 0;
            end
        end
    end
    I2 = logical(I2);
    axis(handles.axes2);
    imshow(I2, 'Parent', handles.axes2);
    axis(handles.axes4);
    c = imhist(uint8(I2));
    bar(c, 'Parent', handles.axes4);
end
```

Слика 21. Button Basic Global Threshold

Функција за *button - Otsu method* је приказана на слици број 22.

```
function otsu_Callback(hObject, eventdata, handles)
I = getappdata(0,'sendData');
I2 = I;
if(I2 == 0)
     clear I;
     f = warndlg('Please, load image!', 'Warning');
else
    hist = imhist(I2);
    sizeImage = size(I2,1) * size(I2,2);
    max = 0;
    for i = 1:256
        P(i) = hist(i) / sizeImage;
    end
    w1 = 0;
    w2 = 0;
    for i = 2:256
        w1 = sum(P(1:i));
        w2 = sum(P(i+1:256));
        mi1 = dot([0:i-1],P(1:i))/w1;
        mi2 = dot([i:255],P(i+1:256))/w2;
        sigma = w1*w2*((mi1-mi2)^2);
        if (sigma > max)
            max = sigma;
            tresh = i - 1;
        end
    end
    for i= 1 : size(I2,1)
        for j = 1 : size(I2,2)
            if (I2(i,j) > tresh)
                I2(i,j) = 1;
            else
                I2(i,j) = 0;
            end
        end
    end
    I3 = I2;
    I2 = logical(I2);
    axis(handles.axes2);
```

Слика 22. Button Otsu method

А, приказ резултата те функције је приказан на слици број 23.

```
axis(handles.axes2);
imshow(I2, 'Parent', handles.axes2);
axis(handles.axes4);
c = imhist(I3);
bar(c, 'Parent', handles.axes4);
```

Слика 23. Button Otsu method приказ

Функција за *button - smoothing* је приказана на слици број 24.

```
% --- Executes on button press in smoothing.
function smoothing_Callback(hObject, eventdata, handles)
I = getappdata(0,'sendData');
I2 = I;
if(I2 == 0)
    clear I;
    f = warndlg('Please, load image!', 'Warning');
else
    I3 = imnoise(I2,'Gaussian',0.04,0.003);
    I3 = double(I3);
    sigma = 1.76;
    sz = 1;
    [x,y]=meshgrid(-sz:sz,-sz:sz);
    M = size(x,1);
    N = size(y,1);
    Exp_comp = -(x.^2+y.^2)/(2*sigma*sigma);
    Kernel= exp(Exp_comp)/(2*pi*sigma*sigma);
    Output=zeros(size(I3));
    %Pad the vector with zeros
    I3 = padarray(I3,[sz sz]);
    %Convolution
    for i = 1:size(I3,1)-M
        for j =1:size(I3,2)-N
            Temp = I3(i:(i+M)-1,j:(j+M)-1).*Kernel;
            Output(i,j)=sum(Temp(:));
        end
    end
    %Image without Noise after Gaussian blur
    Output = uint8(Output);
```

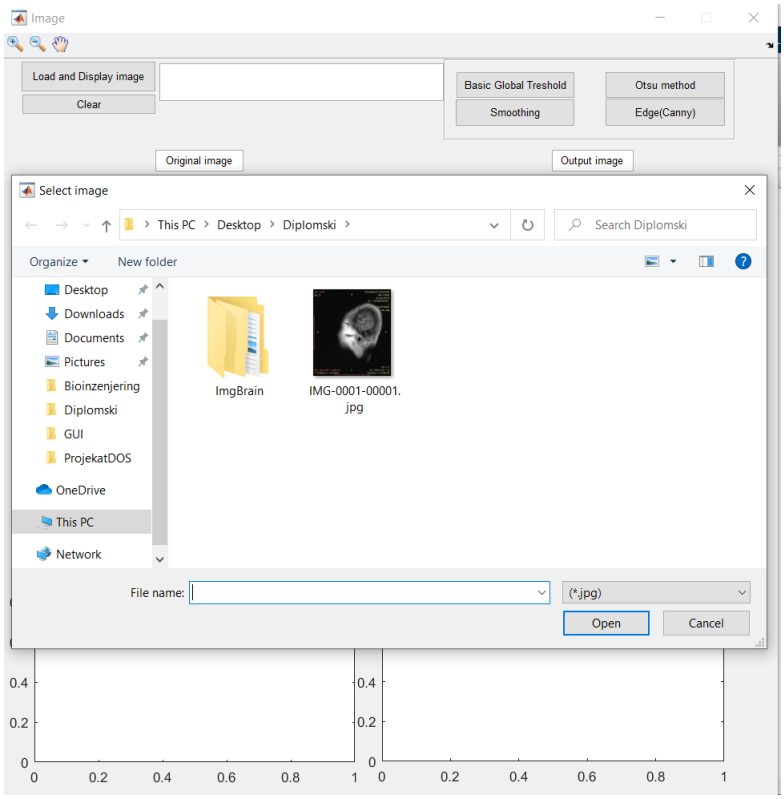
Слика 24. Button Smoothing

Други део кода који се користи за примену *threshold*-а јесте исти као на слици 20. Због обимности кода за *button - Canny* биће приказан само почетак реализације, а остатак кода се може видети у оквиру секције 3.4

```
% --- Executes on button press in canny.
function canny_Callback(hObject, eventdata, handles)
I = getappdata(0,'sendData');
I2 = I;
if(I2 == 0)
    clear I;
    f = warndlg('Please, load image!', 'Warning');
else
    I2 = double (I2);
    %Value for Thresholding
    T_Low = 0.05;
    T_High = 0.12;
    I3 = I2;
    I3 = double(I3);
    sigma = 1.76;
    sz = 1;
    [x,y]=meshgrid(-sz:sz,-sz:sz);
    M = size(x,1);
    N = size(y,1);
    Exp_comp = -(x.^2+y.^2)/(2*sigma*sigma);
    Kernel= exp(Exp_comp)/(2*pi*sigma*sigma);
    Output=zeros(size(I3));
    %Pad the vector with zeros
    I3 = padarray(I3,[sz sz]);
    %Convolution
    for i = 1:size(I3,1)-M
        for j =1:size(I3,2)-N
            Temp = I3(i:(i+M)-1,j:(j+M)-1).*Kernel;
            Output(i,j)=sum(Temp(:));
        end
    end
    %Image without Noise after Gaussian blur
    Output = uint8(Output);
```

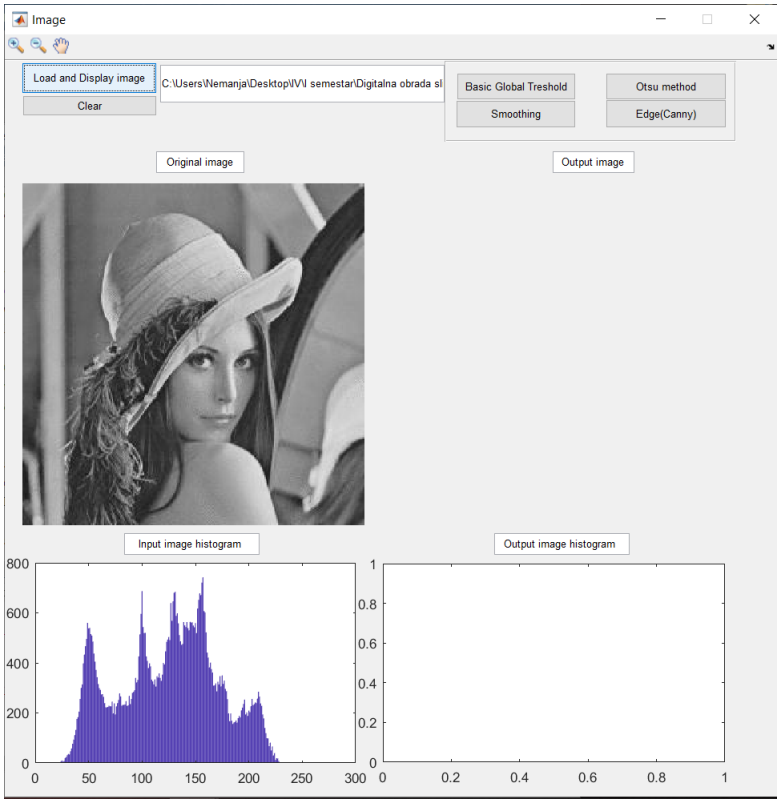
Слика 25. Button Canny

Кликом на *button - Load and Display image* отвара се нов прозор који омогућава претрагу фајлова на рачунару који ће бити учитан (слика 26).



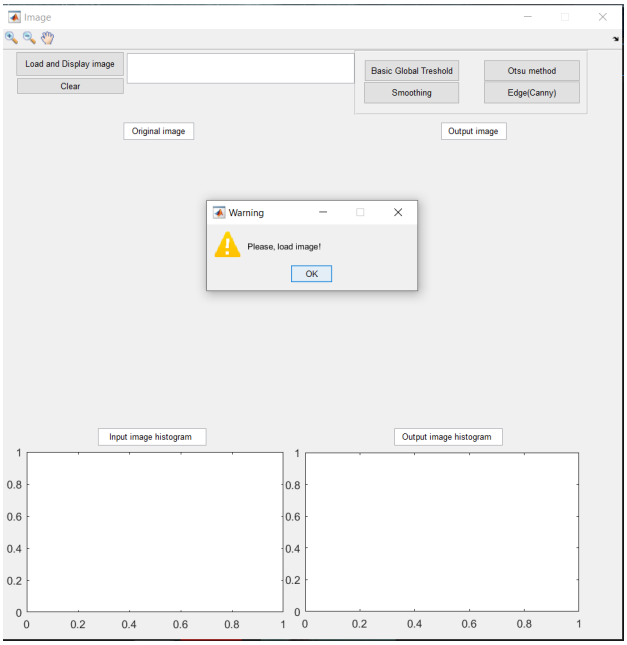
Слика 26. Учитавање слике

Након избора жељене слике и одабира опције *Open*, изабрана слика се учитава у апликацију, приказује се на левој половини као и њен хистограм и путања до тог фајла (слика 27).



Слика 27. Приказ слике и њеног хистограма

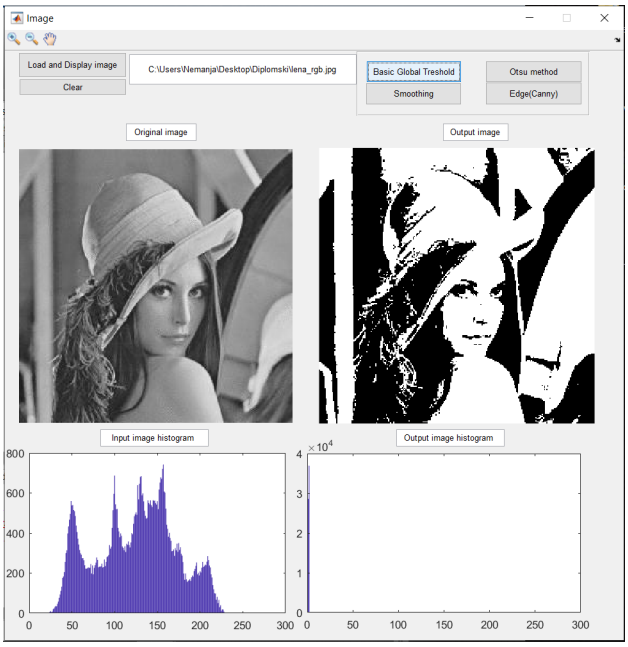
Када слика није изабрана, односно учитана и када се покуша избор неке од опција добија се обавештење које говори да је потребно прво учитати слику (слика 28). Када се учита жељена слика потребно је изабрати неку од опција које ће извршити одређену функцију. У оквиру секције 4, биће приказан начин реализације и резултати након примене.



Слика 28. Обавештење када слика није учитана

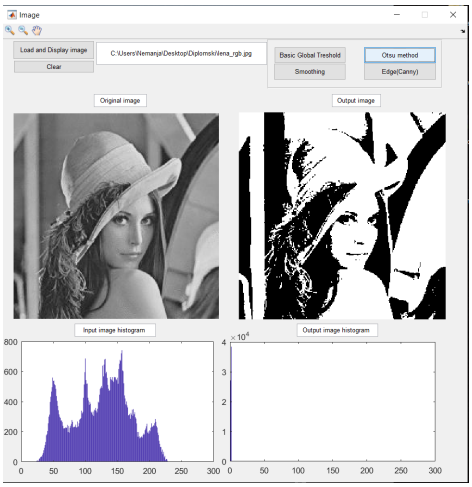
4 Приказ резултата

У оквиру ове секције биће дат приказ примера на којима су примењене функције. На почетку биће приказан пример који је наведен у секцији 3.5 (слика 27). Након примене функције за одређивање глобалног прага која се базира само на интензитету добија се резултат приказан на слици број 29.



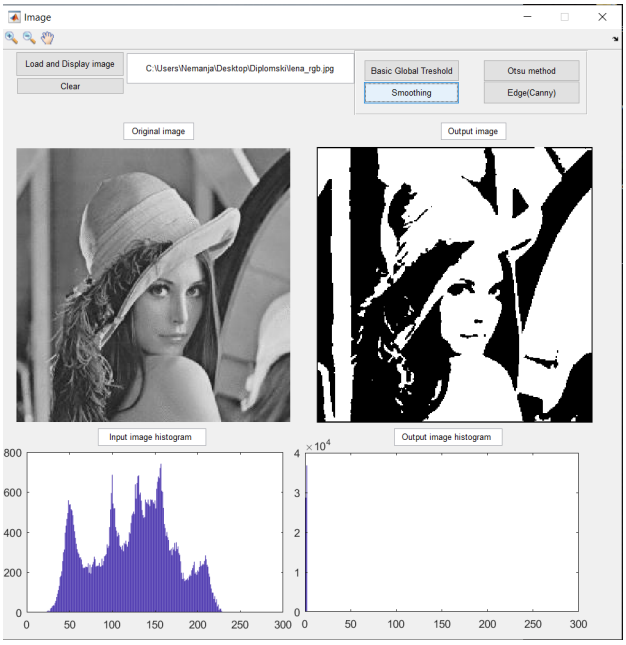
Слика 29. Примена методе у оквиру button-a Basic Global Treshold

Уочава се да је у већој мери функција на добар начин одвојила објекат од позадине. Резултат добијен *Otsu* методом је приказана на слици број 30.



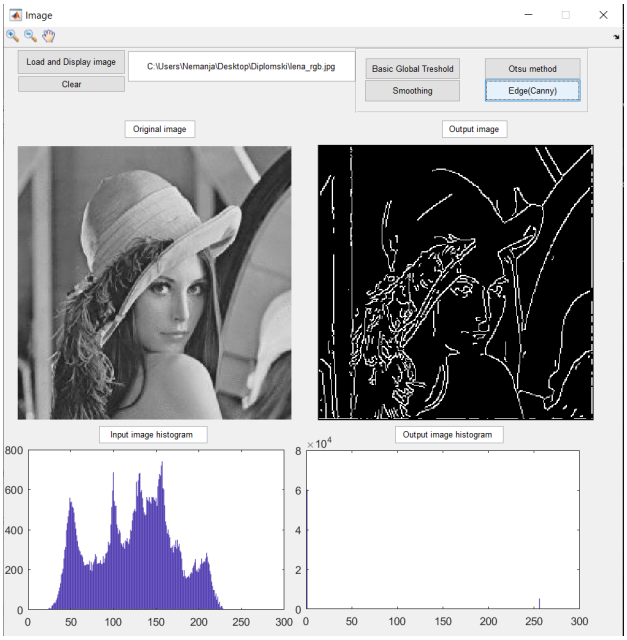
Слика 30. Примена Otsu методе на Лена слици

Анализом ове две методе се уочава да је резулатат у великој мери идентичан, јер је разлика вредности за глобални праг мала. Када се примени метода у којој се примени филтрирање, уочава се да је дошло до губитка информација, неки детаљи су изгубљени, али је циљ испуњен, објекат је издвојен од позадине на добар начин (слика 31).



Слика 31. Резултат након филтрирања и примене treshold-а на Лена слици

Примена Canny алгоритма је детектовала добар део ивица који су дефинисали објекат(слика 32).



Слика 32. Резултат након Canny методе на Лена слици

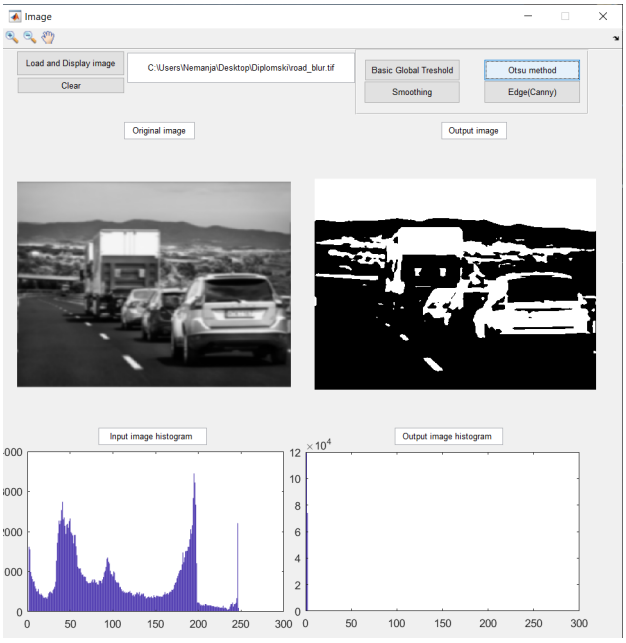
Следећи пример који ће бити разматран јесте упоредна анализа слике која је замућена и слике без замућена. Прва упоредна анализа ће бити у оквиру функције која је реализована на основу рачунања средње вредности. Прва слика јесте слика која је замућена, слика број 33.

Слика 33. Замућена слика, примена методе са средњом вредношћу за праг

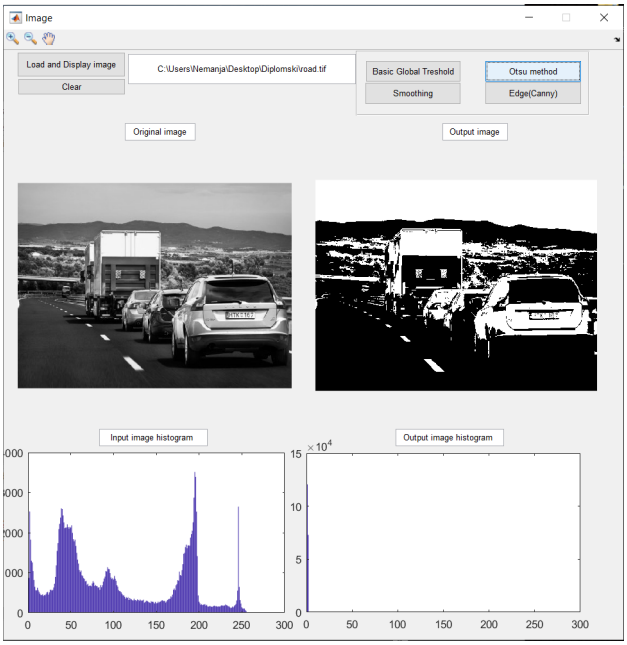
А, слика 34. представља слику која није замућена.

Слика 34. Слика без замућења, примена методе са средњом вредношћу за праг

Разлика која је присутна се огледа у детаљима који су потпунији код слике код које нема замућења, што је и очекивано због јаснијих разлика између компоненти у слици. Следећа анализа је у оквиру *Otsu* методе, прва слика представља слику са замућењем, а друга је без.

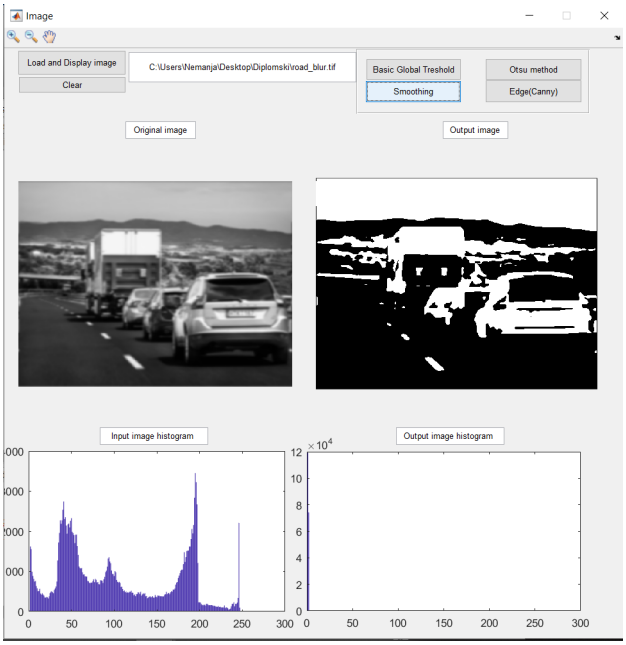


Слика 35. Слика са замућењем, примена Otsu методе



Слика 36. Слика без замућења, примена Otsu методе

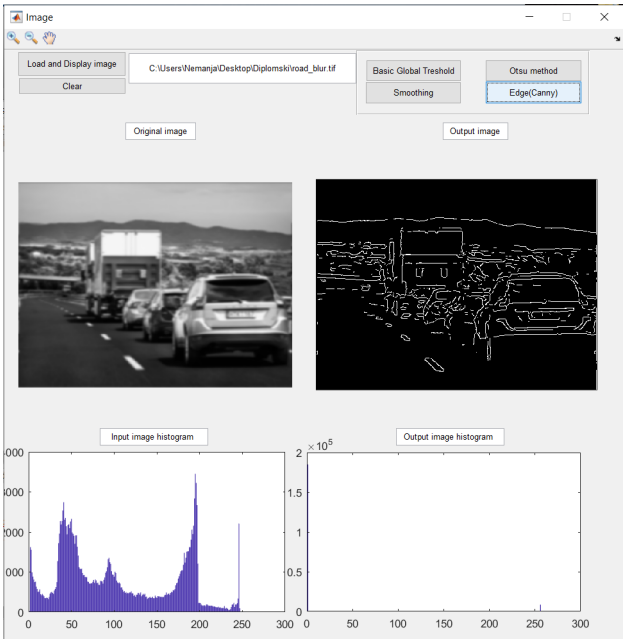
Разлика која је присутна је из истих разлога као код слика 32 и 33. Применом филтрирања су постигнути следећи резултати.



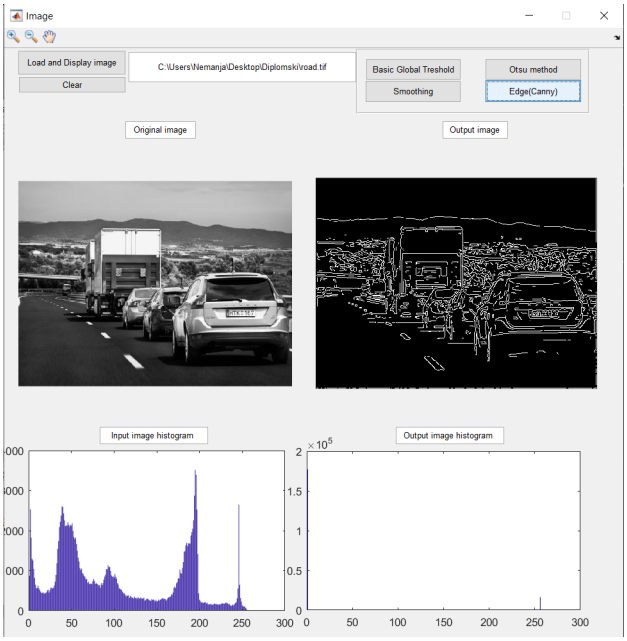
Слика 37. Слика са замућењем, примена филтрирања

Слика 38. Слика без замућења, примена филтрирања

Резултати добијени после примене филтрирања су слични резултатима добијеним применом првих двеју метода. Следећи резултати јесу резултати добијени након примене *Canny* методе.

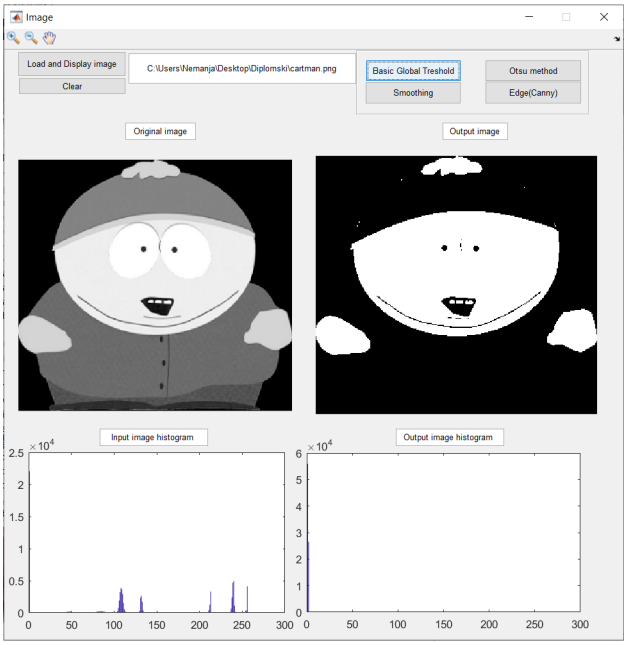


Слика 39. Слика са замућењем, примена Canny методе



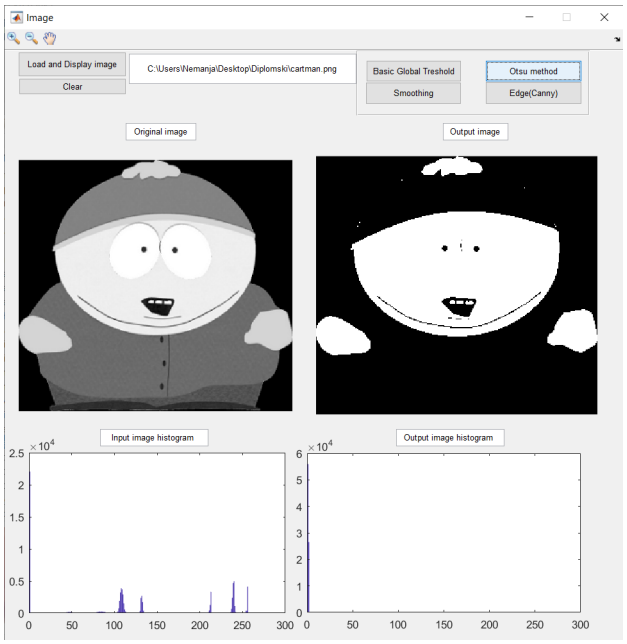
Слика 40. Слика без замућења, примена Canny методе

Разлика је присутна из већ горе наведених разлога, али се може приметити да су ивице у оквиру (слика 38) доста добро детектоване. Следећи пример који ће бити разматран јесте када слика нема изражен контраст. Први пример јесте резултат након примене методе са средњом вредношћу интензитета (слика 41).



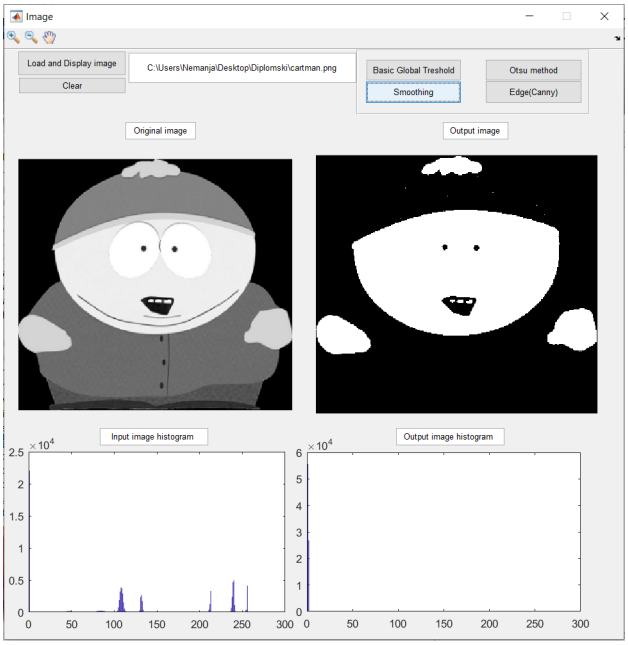
Слика 41. Слика са не тако израженим контрастом

Уочава се да није дошло до добре поделе на објекат и позадину, јер слика сама по себи нема изражен контраст, тако да је дошло до погрешне поделе. Резултат *Otsu* методе је индентичан резултату на слици 41.



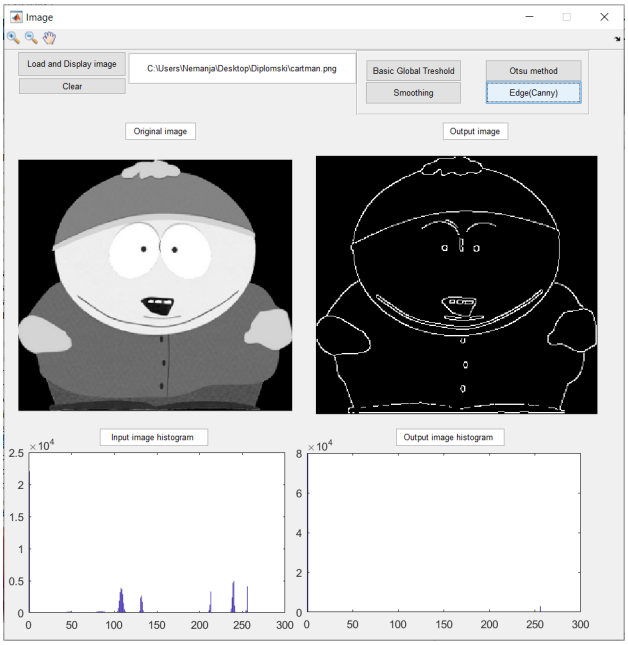
Слика 42. Слика са не тако израженим контрастом, Otsu метода

Како примена методе у којој је присутно филтрирање доводи до губитка информација, на овом примеру је такође присутан губитак информација, поред грешке при *threshold*-у (слика 43).



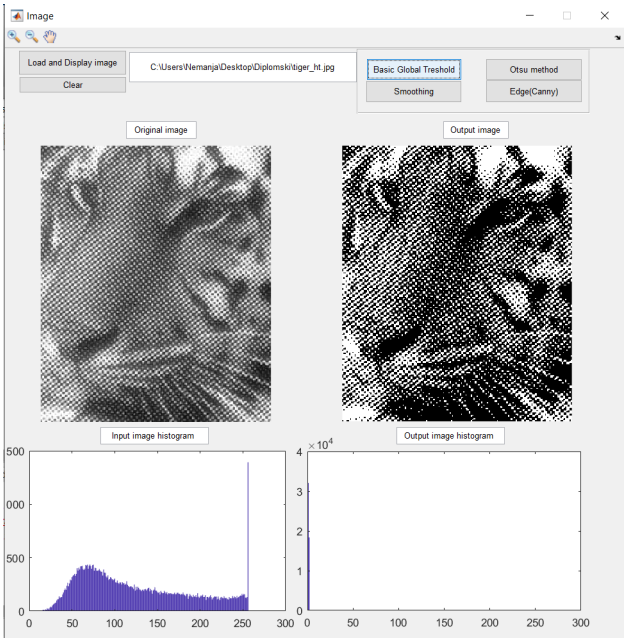
Слика 43. Слика са не тако израженим контрастом, примена филтрирања

Canny метода је приказала неке податке који нису били присутни у претходним примерима, јер се базира на детекцији ивица(слика 44).



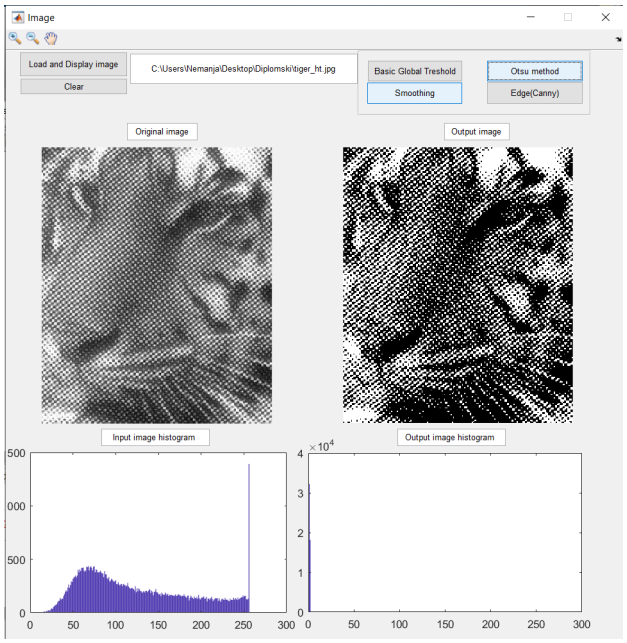
Слика 44. Слика са не тако израженим контрастом, Canny метода

Пример који је следећи обрађен је када је на улаз доведена слика са шумом. Резултат методе са средњом вредношћу интензитета је приказан на слици број 45.



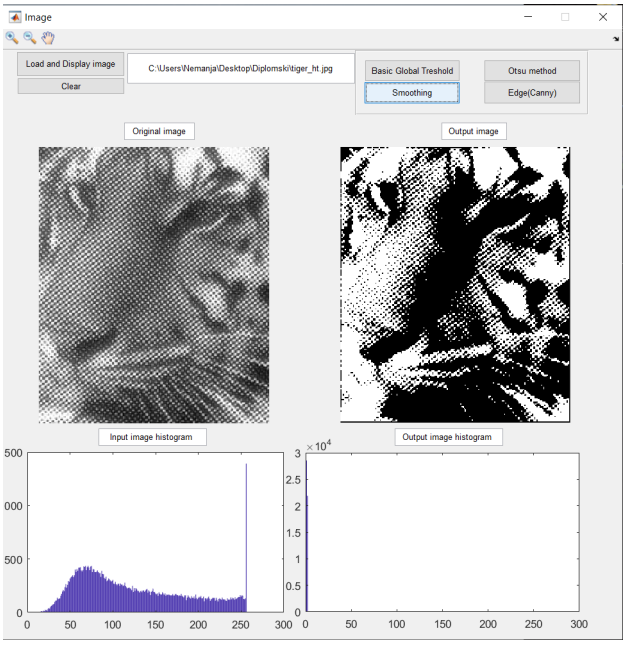
Слика 45. Слика са шумом

Резултат је такав да се облик уочава, али се примећује и нека текстура по слици која је узрокована шумом. *Otsu* метода да је исти резултат (слика 46).



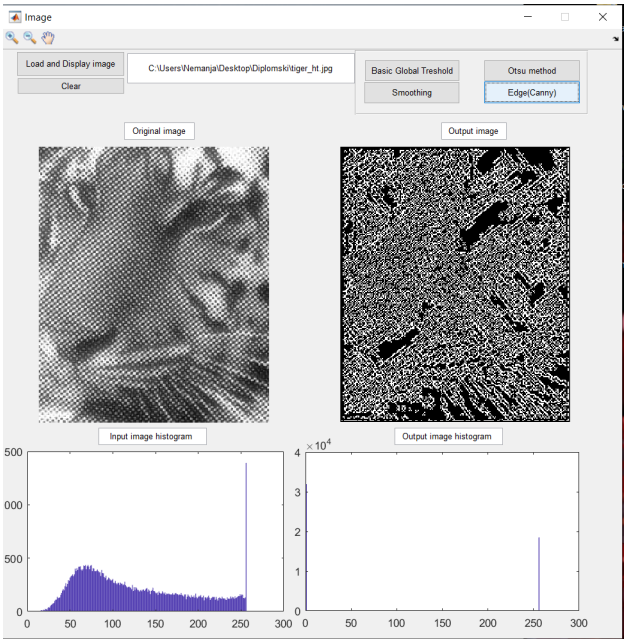
Слика 46. Слика са шумом, Otsu метода

Примена филтрирања доводи до тога да је да се текстура која је присутна губи, али је присутно и губљење јасноће објекта (слика 47), разлози су већ наведени у претходним примерима.



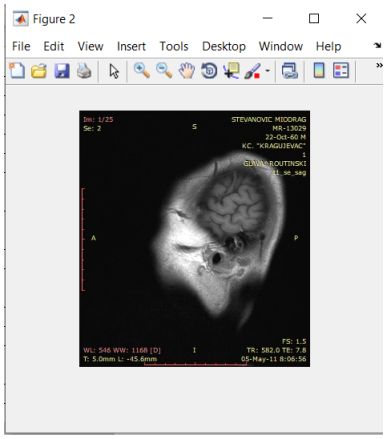
Слика 47. Слика са шумом, примена филтрирања

Због присуства текстуре дошло је до лажног детектовања ивица, тако да је резултат такав због великог утицаја шума (слика 48).



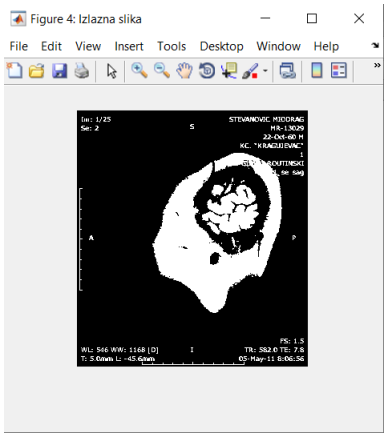
Слика 48. Слика са шумом, Canny

Пример који је реализован јесте када је примењена *Canny* метода на слику која је претходно процесирана *Otsu* методом и исте те слике пре него што је примењен *Canny* алгоритам. Улазна слика је:



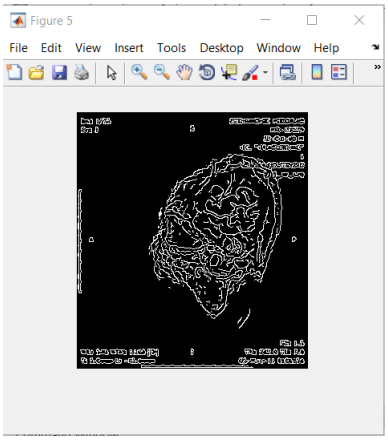
Слика 49. Улазна слика

Када се примени *Otsu* метода добија се следећа слика:



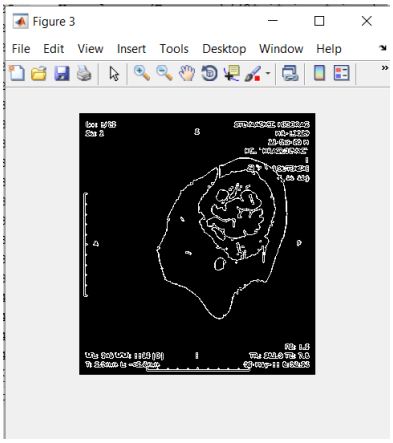
Слика 50. Улазна слика након примене Otsu методе

Резултат *Canny* алгоритма на оригиналну слику јесте:



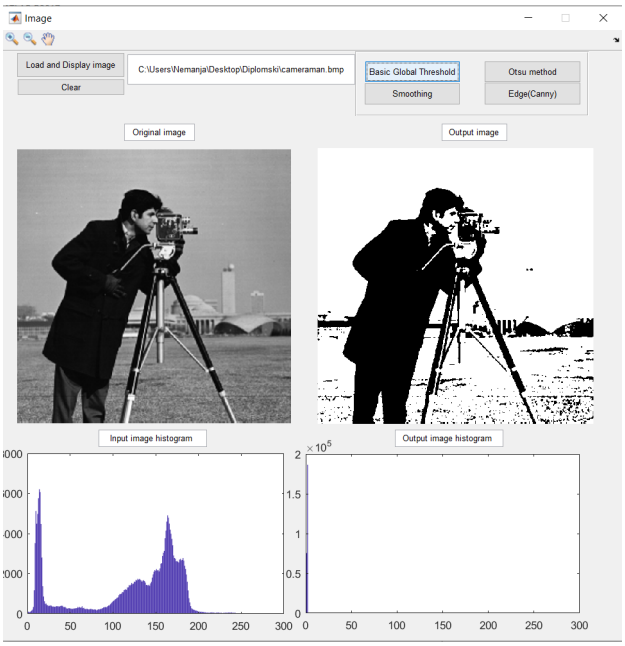
Слика 51. Улазна слика након примене Canny методе

А, резултат *Canny* алгоритма примењеног на слику на којој је примењен *Otsu* јесте:

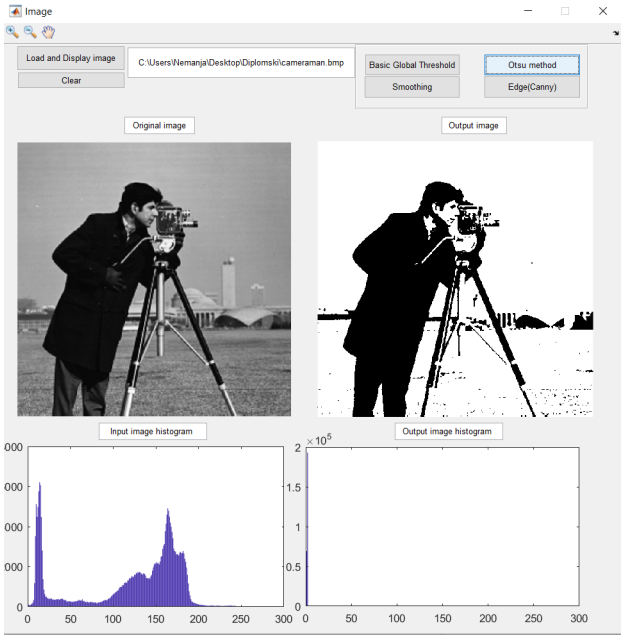


Слика 52. Улазна слика са глобалним treshold-ом (Otsu) након примене Canny методе

Оно што се уочава јесте јаснија облика који је присутан на слици 52. Како је резултат глобалног *threshold*-а одвојио објекат, и у оквиру њега један део дефинисао униформно омогућено је јасније дефинисање, у овом примеру, мозга. За раулику од слике 51 где није баш могуће јасно идентификовати мозак, због присуства остатка ивица. Ово показује да је пре-процесирање података веома битна ставка за решавање проблема сегементације и детекције ивица. Сви примери који су до сада представљени су имали веома приближне вредности за праг. На следећем примеру је уочљива разлика између оптималног и глобалног прага.



Слика 53. Слика са глобалним прагом



Слика 54. Слика са оптималним прагом

Разлика у виду позадине показује разлике између глобалног и оптималног прага.

5 Закључак

Примери који су коришћени су изабрани тако да покрију што више ситуација са којима се може сусрести приликом обраде. Задатак јесте био упоређивање резултата који су добијени практичном имплементацијом и очекиваних резултати који су наведени приликом теоријског описа сваког сегмента. Анализом резултата који су доби-јени практичном имплементацијом се може увидети потврда очекиваних резултата, који зависе од проблема који је потребно решити. Прецизност датих резултата је веома битна, јер примена ових метода има веома широку употребу и простор за погрешне вредности није дозвољен. На основу резултата добијених долази се до закључка да је у неким ситуацијама могуће постићи довољно добро решење применом једноставног метода оптималног глобалног прага, али у неким и није. На основу примера који су коришћени за тестирање, уочава се да резултат који је добијен применом глобалног прага може бити приближан резултату који је добијен оптималним прагом. Али постоје примери у којима је разлика евидентна, и где примена оптималног прага да је боље резултате, случај последњег примера. Недостатак *Otsu* методе се може видети на примеру приказаном на слици број 42. Како би се тај проблем решио потребно је увођење још једне вредности за праг. Увођењем још једне вредности се рачунски компликује одређивање прагова. Ако слика садржи велики број потенцијалних прагова које је потребно пронаћи, метода постаје све сложенија и захтевнија и потребно је пронаћи друге начине реализације.

6 Литература

1. [1] Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
2. (1) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
3. (2) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
4. (3) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
5. (4) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
6. (5) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
7. (6) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
8. (7) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
9. (8) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
10. (9) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
11. (10) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
12. (11) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
13. (12) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
14. (13) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
15. (14) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
16. (15) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
17. (16) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
18. [2] Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
19. [3] Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall
20. [4] Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall

21. (17) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Ha

22. [5] Преузето: `https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm` датум приступа 13.09.2020 13:00 PM

23. (18) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Ha

24. [6] Преузето:https://www.researchgate.net/figure/Image-convolution-with-an-input-image-of-size-7-7-and-a-filter-kernel-of-size-3-3_fig1_318849314 датум приступа 13.09.2020 18:00 PM

25. [7] Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall

26. [8] Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall

27. [9] Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall

28. [10] Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall

29. (19) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Ha

30. (20) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Ha

31. (21) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Ha

32. (22) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Ha

33. (23) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Ha

34. (24) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Ha

35. (25) Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Ha

36. [11] Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall

37. [12] Преузето: Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing 3rd Edition* (2008) Prentice Hall