**UNIVERSITETI I PRISHTINËS**

**FAKULTETI I SHKENCAVE MATEMATIKE NATYRORE**

DEPARTAMENTI I MATEMATIKËS

Programi: Shkencë Kompjuterike



Lënda: Procesimi i imazheve

Detyra 1

Profesori: Studentët:

Besnik Duriqi Fjolla Sellimi

Erëza Asllani

Fatjonë Thaçi

Prill 2023

**Përmbajtja**

[**Lista e figurave** 2](#_Toc135691781)

[**Përshkrimi i detyrës** 3](#_Toc135691782)

[**Hapat e kodit:** 3](#_Toc135691783)

# **Lista e figurave**

[Figura 1. 4](#_Toc135691758)

[Figura 2. 4](#_Toc135691759)

[Figura 3. 5](#_Toc135691760)

[Figura 4. 6](#_Toc135691761)

[Figura 5. 7](#_Toc135691762)

[Figura 6. 7](#_Toc135691763)

[Figura 7. 8](#_Toc135691764)

[Figura 8. 8](#_Toc135691765)

[Figura 9. 9](#_Toc135691766)

[Figura 10. 10](#_Toc135691767)

[Figura 11. 10](#_Toc135691768)

[Figura 12. 11](#_Toc135691769)

# **Përshkrimi i detyrës**

Kjo detyrë do të ngarkojë një imazh të caktuar dhe do të kryejë disa operacione në të për të demonstruar disa nga funksionet e imazheve në OCTAVE. Së pari, imazhi ngarkohet dhe shfaqet përmes funksionit imshow(). Pastaj, ai është konvertuar në një imazh bardh e zi përmes funksionit rgb2gray(). Pastaj janë kryer disa operacione filtrimi në imazhin bardh e zi duke përdorur funksionin imfilter(). Në veçanti, janë përdorur një filtër gaussian, një filtër sobel dhe një filtër median. Përveç kësaj, një filtër i veçantë është implementuar manualisht dhe aplikuar në imazhin bardh e zi. Pastaj është bërë barazim i histogramit nëpërmjet funksionit histeq(). Në fund, disa operacione morfologjike janë kryer në një imazh binar duke përdorur funksionet erode(), dilate(), open() dhe close() me elemente strukturore të ndryshme strel(). Ky kod është një shembull i thjeshtë për të kuptuar si OCTAVE mund të përdoret për të kryer disa nga funksionet e imazheve të zakonshme.

# **Hapat e kodit:**

Së pari, shkruajmë komandën :

>> pkg load image

Kjo komandë ngarkon paketën e imazheve dhe pa të funksionet e imazheve nuk do të jenë të disponueshme.

Pastaj për të ngarkuar imazhin përdorim komandat:

>>img = imread(“Desktop/microscopy\_image.jpg”);

>>imshow(img);

Ekzekutimi i këtyre komandave na shfaq figurën e mëposhtme:

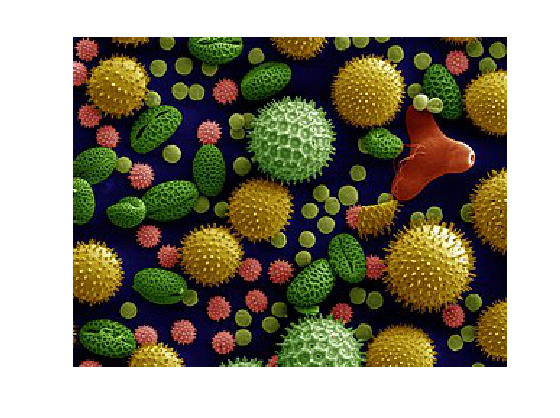


Figura 1.

Pastaj kemi komandën:

>>gray\_img = rgb2gray(img);

Kjo komadë e kthen figurën bardh e zi. Figura është e paraqitur më poshtë.

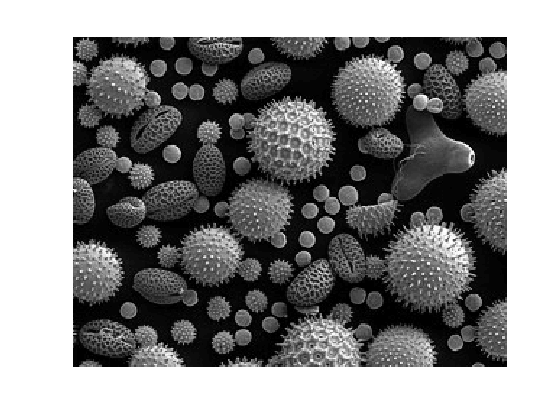


Figura 2.

Pastaj ekzekutojmë komandat:

>>gaussian\_filter = fspecial("gaussian", [5 5], 1);

>>gaussian\_img = imfilter(gray\_img, gaussian\_filter);

>>imshow(gaussian\_img);

Komanda "fspecial("gaussian", [5 5], 1)" kthen një filtrin të tipit Gauss me një madhësi prej 5x5 dhe një devijim standard prej 1. Kjo do të thotë se ky filtër do të zbresë intensitetin e ndryshme të ngjyrave në imazhin e dhënë, duke krijuar një efekt të dobësimit të imazhit në një mënyrë të ndjeshme ndaj shumëzimit të tij me një masë të dytë. Pastaj, komanda "imfilter(gray\_img, gaussian\_filter)" përdor këtë filtrin për të filtruar imazhin grayscale "gray\_img". Kjo do të thotë se secili pixel në imazh do të zëvendësohet me një vlerë të përcaktuar nga filtri i Gaussit. Rezultati shfaqet në figurën 3.

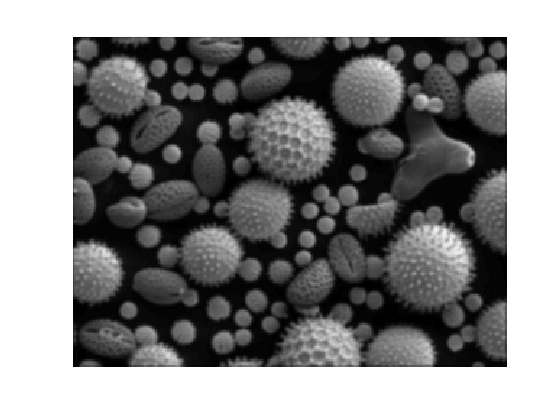


Figura 3.

Kemi përdorur edhe filtrin sobel me anë të komadave:

>>sobel\_filter = fspecial("sobel");

>>sobel\_img = imfilter(gray\_img, sobel\_filter);

>>imshow(sobel\_img);``

Filtri Sobel përdoret për deteksionin e kufijve në imazhe. Ai përdoret për të identifikuar kufijtë vertikalë dhe horizontalë në një imazh duke përdorur dy filtrat e ndryshme, një për secilin drejtim. Përdorimi i këtij filtri në OCTAVE është i ngjashëm me filtrin e Gaussit. Rezultati shihet në figurën 4.

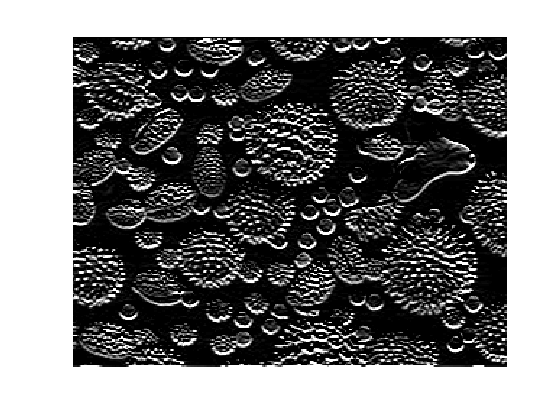


Figura 4.

Pastaj për të hjekur zhurmat përdorim filtrin median.

>>median\_img = medfilt2(gray\_img, [3 3]);

>>imshow(median\_img);

Filtri median është një lloj filtri që përdoret për reduktimin e zhurmave në një imazh. Ky filtri zëvendëson vlerën e pikselave në imazh me medianën e vlerave të pikselave të një regjioni të caktuar rreth tij. Në këtë rast, komanda "medfilt2" aplikohet në imazhin e grumbulluar "gray\_img" dhe kërkon një matricë filtrash [3 3] për të ndryshuar çdo piksel në imazh me medianën e vlerave të pikselave brenda një regjioni 3x3 që rrethon atë. Ky proces ndihmon në reduktimin e zhurmave të vogla, të shpeshta në imazh.

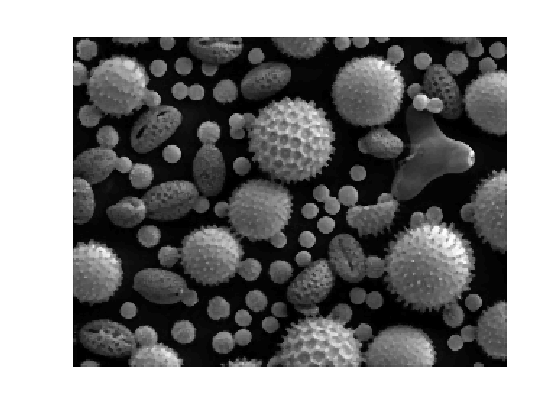


Figura 5.

Pastaj kemi aplikuar një filtër të personalizuar me anë të komandave:

>> custom\_filter = [-1 -1 -1; -1 8 -1; -1 -1 -1];

>>custom\_img = imfilter(gray\_img, custom\_filter);

>>imshow(custom\_img);

Ky kod implementon një filtrin të personalizuar, ku matrica e filtrit është [-1 -1 -1; -1 8 -1; -1 -1 -1]. Ky filtër i quajtur "filtri i ndjeshëm i kanterës" (ang. "edge-sensitive filter") mund të shënojë vijat e ndryshme ose ndryshime drastike në intensitetin e ngjyrës brenda një imazhi.

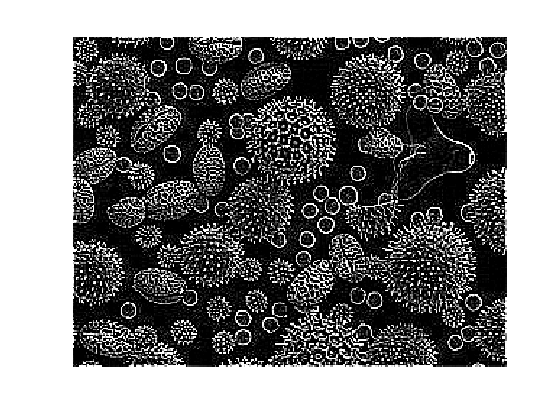


Figura 6.

Përveç aplikimit të këtyre filterave, kemi aplikuar edhe barazimin e histogramit me anë të komandave:

>> eq\_img = histeq(gray\_img);

>>imshow(eq\_img);

Dhe rezultati shihet në figurën 7.

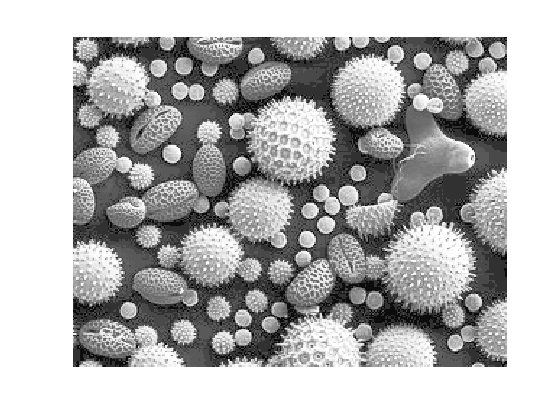


Figura 7.

Pastaj kemi aplikuar edhe operacionet morfologjike në imazhin binar.

>> binary\_img = im2bw(gray\_img);

>>imshow(binary\_img);

Këto komanda e bëjnë binarizimin e imazhit, thresholding për të ndarë pikselat e errët nga ata të ndriçuar. Rezultati shihet në figurën 8.

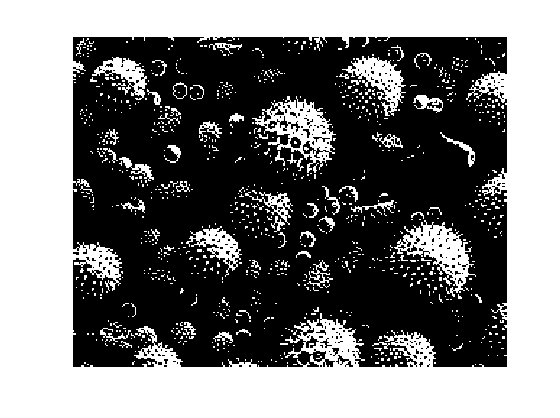


Figura 8.

Pastaj, meqë komanda strel(“disk”, 3); nuk është e disponushme në Octave duhet të shkrujamë këtë kod për të arritur te rezultati i dëshiruar.

>>radius = 3;

>>diameter = 2 \* radius + 1;

>>[x, y] = meshgrid(-(radius):(radius));

>>se\_disk = ((x.^2 + y.^2) <= radius^2);

>>se = strel("arbitrary", se\_disk);

Këto komanda përdoren në OCTAVE për të krijuar një element strukturore binare me formën e një diskut dhe pastaj krijojnë një element strukturore binare "arbitrary" që përdoret për morfologjinë e imazheve.

Aplikimi i operacioneve morfologjike si :

>>eroded\_img = imerode(binary\_img, se);

>>imshow(eroded\_img); //Rezultati Figura 9

>>dilated\_img = imdilate(binary\_img, se);

>>imshow(dilated\_img); //Rezultati Figura 10

>>opened\_img = imopen(binary\_img, se);

>>imshow(opened\_img); //Rezultati Figura 11

>>closed\_img = imclose(binary\_img, se);

>>imshow(closed\_img); //Rezultati Figura 12



Figura 9.

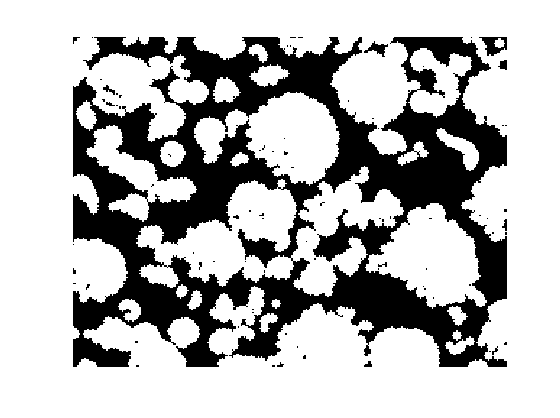


Figura 10.



Figura 11.

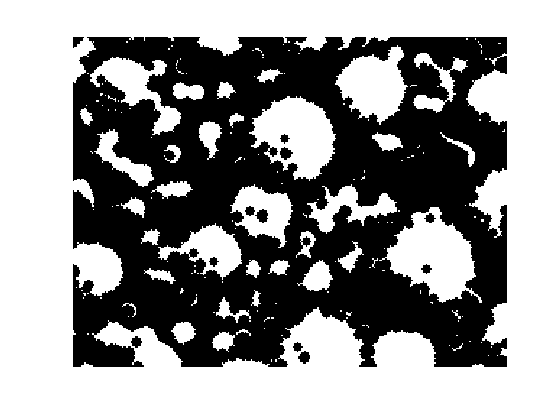


Figura 12.