

Academia de Studii Economice din București

Facultatea de Cibernetică, Statistică și Informatică Economică

Specializarea Informatică Economică

Tehnici de procesare a imaginilor

Documentația completă a proiectului

Coordonatori:

Prof.univ.dr Cătălina COCIANU

Prof.univ.dr Cristian USCATU

Student

Ioana-Diana PETRE

București 2021

Cuprins

Contents

[Temele abordate 3](#_Toc60234148)

[Partea I 3](#_Toc60234149)

[Partea II 3](#_Toc60234150)

[Partea III 4](#_Toc60234151)

[Partea IV 4](#_Toc60234152)

[Partea V 5](#_Toc60234153)

# Temele abordate

În acest capitol sunt descrise temele alese, modul de implementare pentru acestea, datele de intrare necesare, rezultatele obținute și exemple de apel pentru a putea rula sursele.

Fiecare sursă prezintă funcții pentru citirea unei imagini și afișarea acesteia cu un titlu ales. Imaginile cu care se lucrează sunt monocrome.

## Partea I

Implementarea reprezentării Karhunen-Loeve pentru seturi de imagini de dimensiune mai mare sau egală de 250 x 250. Imaginile sunt împărțite în blocuri de dimensiune egală.

## Partea II

Implementarea procedurii unsharp masking.

Construim o mască, dimensiunea este dată ca parametru. Inițial masca se va inițializa cu valoarea 1 apoi vom aduna elementele acestei matrice. Vom împărți elementele inițiale ale măști cu suma calculată anterior.

Descrierea algoritmului:

* Se citește imaginea inițială și se convertește către una grayscale dacă este cazul,
* Construim masca de dimensiunea dată,
* Construim o imagine nivelată apicând un filtru de tipul medie asupra imaginii inițiale,
* Procedura de unsharp masking rezultă din calculul imaginii inițiale – imaginea nivelată + imaginea inițială,
* Afișăm rezultatul obținut în urma acestui calcul.

S-a observat că pentru o imagine inițială perturbată rezultatele sunt foarte bune. Pentru o imagine inițială clară nu putem observa atât de bine beneficiile algoritmului.

Funcția ***unsharp\_masking( img, dimensiune\_masca )*** implemenetează algoritmul descris anterior, unde ***img*** este imaginea inițială iar ***dimensiune\_masca*** reprezintă dimensiunea măștii pentru care aplicăm filtrul de nivelare medie.

Exemple de rulare:

* unsharp\_masking('img.png', 11);
* unsharp\_masking('img.png', 5);
* unsharp\_masking('img\_perturb.bmp', 5);

## Partea III

Implementarea filtrului trece-jos gaussian.

Descrierea algoritmului:

* Se citește imaginea inițială și se convertește către una grayscale dacă este cazul,
* Imaginea inițială trebuie expandată având de două ori mai multe linii și coloane. Este important să reținem centrul acesteia înainte de expandare,
* Se va centra imaginea expandată,
* Imaginea centrată o vom transforma în domeniul frecvențelor,
* Construim filtrul H utilizând următoarea formulă: , unde D0 este o rază dată a filtrului trece-jos gaussian iar reprezintă distanța de la (n,m) la centrul de simetrie al imaginii calculat la început,
* Se va calcula matricea G care rezultă prin înmulțirea element cu element a imaginii centrate în domeniul frecvențelor și filtrul trece-jos gaussian,
* Reconstruiesc imaginea filtrată revenid în domeniul spatial,
* Elimin centrarea asupra imaginii reconstruite,
* Extrag rezultatul dorit revenind la dimensiunile imaginii inițiale,
* Afișez rezultatul.

Funcția ***filtru\_low\_pass\_gauss(img, D0)*** implementează acest algoritm, unde ***img*** reprezintă imaginea inițială și ***D0*** raza filtrului gaussian.

Folosind o imagine de dimensiuni mai mici, s-a observat că cu cât raza este mai mare cu atât imaginea rezultată va fi mai fin nivelată.

Exemple de apel:

* filtru\_low\_pass\_gauss('img.png', 50);
* filtru\_low\_pass\_gauss('img.png', 30);

## Partea IV

Implementarea filtrului median adaptiv. Acest tip de filtrare elimină zgomotul de tip impuls (sare, piper) cu o probabilitate de apariție mare și păstrează detalii de tip contur.

Algoritmul extinde imaginea perturbată folosind o mască de dimensiune impară. La început masca va avea dimensiunea 3x3. Extinderea imaginii se face adăugând (dimensiunea măștii curente + 1) / 2 elemente de 0 la frontierele.

Vom extrage o mască curentă pentru fiecare pixel de dimensiune d, apoi aceasta se va liniariza și sorta pentru a putea obține valoarea minima, maximă și mediană din șirul de gri curent.

Indicii alpha1, alpha2, beta1 și beta2 sunt calculați pe baza valoriilor obținute anterior astfel:

* alpha1 = val\_mediana - val\_min;
* alpha2 = val\_mediana - val\_max;
* beta1 = fc(i, j) - val\_min;
* beta2 = fc(i, j) - val\_max;

fc(i, j) reprezintă valoarea pixelului din imaginea perturbată de la coordonatele (i,j).

Descrierea algoritmului:

* Se citește imaginea inițială și se convertește către una grayscale dacă este cazul,
* Se va perturba imaginea inițială cu un zgomot de tip impuls având o probabilitate de apariție mare,
* Parcurgem fiecare pixel al imaginii perturbată,
* Pentru fiecare pixel definim inițial dimensiuea măștii ca fiind 3 și extindem imaginea,
* Se obține o mască curentă și o folosim pentru a calcula inițial indicia alpha1, alpha2,
* Dacă alpha1 > 0 și alpha2 < 0, atunci vom calcula indicia beta1 și beta2,
* Dacă beta1 > 0 și beta2 < 0 atunci vom păstra valoarea pixelului de la coordonatele curente în imaginea filtrată, altfel vom păstra valoarea mediană corespunzătoare șirului de gri curent (al măștii),
* Dacă alpha1 și alpha2 nu respectă condiția impusă mai sus vom extinde dimensiunea măștii,
* Acești pași vor fi reluați pentru fiecare pixel în parte atâta timp cât dimensiunea măștii nu depășește un prag stability.
* După ce acești pași sunt parcurși pentru fiecare pixel vom afișa imaginea rezultată în urma filtrării.

S-a observat că cu cât mărim pragul dimensiunii măștii cu atât imaginea rezultată va fi din ce în ce mai blurată.

Funcția ***filtrare\_median\_adaptiva( img, d\_max )*** implementează algoritmul descris mai sus, unde ***img*** reprezintă imaginea inițială iar ***d\_max*** dimensiunea maximă pentru care putem extinde masca.

Exemple de apel:

* filtrare\_median\_adaptiva(’img.png’ , 7)
* filtrare\_median\_adaptiva(’img.png’ , 11)

## Partea V

Implementarea filtrului invers în domenioul frecvențelor. Filtrul invers se aplică în cazul motion blur în varianta continua pe axa OX (sau OY).