TREBALL FINAL DE GRAU: CREACIÓ GRÀFICA (MEMÒRIA)

▷ PAC1: Definició formal del projecte

▷ PAC2: Desenvolupament i consolidació I

▷ PAC3: Desenvolupament i consolidació II

▶ PAC4: Lliurament final

Joan Mora Peralta

26/12/2020

Índex

1. Abstract 5
2. Introducció 5
3. Descripció 8
4. Objectius 8
5. Marc teòric 9
5.1 Antecedents 9
5.2 Escenari 9
5.3 Bases teòriques de referència 10
5.3.1 Sensibilitat ISO 10
5.3.2 Estimació del soroll 11
5.3.3 Percepció 11
5.4 Altres estudis similars 11
6. Metodologia 12
6.1 Etapes 12
6.2 Recerca 13
6.2.1 Mètodes 15
6.2.1.1 Mètodes d'eliminació de soroll 16
6.2.1.1.1 Gaussian Bilateral 16
6.2.1.1.2 Crimmins Speckle Removal 17
6.2.1.1.3 Non-Uniform Weight 18
6.2.1.1.4 Bilateral 18
6.2.1.1.5 Gaussian 19
6.2.1.1.6 Conservative 20
6.2.1.1.7 Salt and Pepper Median 20
6.2.1.1.8 Average 21
6.2.1.1.9 Progressive Median 21
6.2.1.1.10 Progressive Conservative 22
6.2.1.1.11 Extreme Median 22
6.2.1.2 Mètodes d'addició de soroll 23

6.2.2 Imatges originals....23
6.3 Anàlisi i conclusions....24
6.3.1Metodologia....24
6.3.2 Valoracions....25
6.4 Investigació....26
7. Planificació....26
8. Arquitectura de l'aplicació....30
9. Plataforma de desenvolupament....31
10. Prototips....31
11. Perfils d'usuari....32
Bibliografia....32
Il·lustracions....32
Fonts....33
Annex: Mètodes....36
Mètode 1: Filtre bilateral gaussià....36
Mètode 2: Crimmins Speckle Removal....37
Mètode 3: Desenfocament amb pes no uniforme....42
Mètode 4: Filtre gaussià....44
Mètode 5: Filtre conservatiu....48
Mètode 6: Eliminació de sal i pebre....49
Mètode 7: Barreja de mediana....51
Mètode 8: Filtre de mediana ampliat per a soroll de sal i pebre....52
Mètode 9: Filtre de mediana....53
Mètode 10: Filtre mitjà....58

1. Abstract

El soroll en les imatges és un fenomen sovint no desitjat en el món de la fotografia i existeixen diversos d'algorismes capaços de reduir-lo, però no tots són igual d'eficaços. En aquest estudi es tractarà d'aportar una mica de llum sobre la manipulació d'aquests mètodes en els píxels, la seva eficàcia i els efectes que aquests ocasionen a les imatges.

Paraules clau: imatges, soroll, algorismes, desenvolupament.

(English version) Noise in images is often an unwanted phenomenon in the world of photography, and there are several algorithms to reduce it, but not all are equally effective. This study will bring some light on the manipulation of these methods on pixels, their effectiveness and the effects they cause on images.

Keywords: images, noise, algoriths, development.

2. Introducció

El soroll, segons DIEC2 de l'Institut d'Estudis Catalans, es pot definir com [1]"3 1 m. [FL] Pertorbació del canal de comunicació que pot ocasionar una pèrdua d'informació al receptor." Que, aplicat en context d'una imatge digital, significa tota aquella representació del color que no correspon amb la realitat. Si partim d'aquest punt, podrem veure que existeixen molts tipus de soroll ocasionats per causes molt diverses, com per exemple el mal funcionament del dispositiu, pertorbacions electromagnètiques en el medi, deficiència de llum durant la captura o alta sensibilitat dels fotodíodes, excloent en aquest treball el tractament d'artefactes o altres fenòmens com el *blooming*. Aquest problema, el soroll, ha existit des de que som conscients dels nostres sentits, però en el cas de la vista humana (per exemple, veient fosfens) és un fet que és alterat pel cervell i ha passat molt inadvertit; després, amb les imatges analògiques, es va fer més present però fins a cert punt no va resultar antiestètic. Ara, entrats en la era digital, aquest no ha estat tan ben tolerat i s'ha convertit en un dels principals problemes per als fotògrafs a l'hora de captar una imatge de qualitat amb la càmera.

A tall d'exemple, podem apreciar alguns d'aquests sorolls de diferents naturaleses, segons la seva causa:

· Soroll de captura: És causat per la granularia dels fotons en el moment de la captura, que dóna lloc a petites diferències aleatòries en la llum enregistrada. Prové també de la naturalesa quàntica de la llum.

Figura 1

MDF. 2010. *Photon-noise.jpg* [imatge]. Wikipedia. Disponible a: https://upload.wikimedia.org/wikipedia/commons/4/42/Photon-noise.jpg

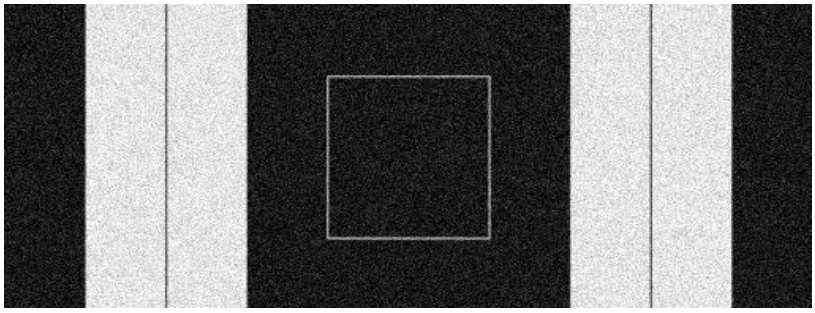
· Soroll impulsional, també anomenat "sal i pebre": És atribuït a la pols o a un mal funcionament dels circuits.

Figura 2

LORENZI, Luca. 2010. *Salt and pepper steps.png* [imatge]. Wikipedia. Disponible a: https://commons.wikimedia.org/wiki/File:Salt_and_pepper_steps.png

· Soroll gaussià: Adopta una distribució gaussiana en la seva probabilitat i s'associa amb la radiació electromagnètica.

Figura 3



Desconegut. 2009. *512x512-Gaussian-Noise.jpg* [imatge]. Wikipedia. Disponible a: https://commons.wikimedia.org/wiki/File:512x512-Gaussian-Noise.jpg

· Soroll cromàtic: No prové de l'entorn i és provocat per la fluctuació de les tonalitats entre píxels.

Figura 4

BACCAGLINI, Fluvio. 2020. *Random Uniform RGB 416x338* [imatge]. Flickr. Disponible a: https://www.flickr.com/photos/185533101@N05/49352561877

Per entendre perquè s'origina hem d'entendre abans com la càmera aconsegueix captar els colors del seu entorn. En resum, els colors són les interpretacions de les diferents freqüències de la llum i aquesta a la vegada està formada per fotons. La càmera capta aquests fotons a través d'uns dispositius sensibles a aquestes pertorbacions, es transformen en senyal elèctrica i després en dades. Entenent aquest esquema és fàcil veure que un sensor brut no captarà els mateixos colors que un en bon estat, per exemple, o que quan una pertorbació electromagnètica desplaça els electrons desviant el seu trajecte cap al sensor, aquest produirà soroll.

Les solucions mai són perfectes, perquè la informació real dels colors es perd i no es pot recuperar, però existeixen diferents mètodes per minimitzar o dissimular els efectes del soroll en les imatges digitals - durant i després de l'enregistrament. En el primer escenari és on el fotògraf té la possibilitat de reduir el soroll, però a vegades els requeriments de l'escena no li permeten la plena llibertat per decidir amb quins paràmetres l'ha de captar. Per tant, serà necessari aplicar després un procés computacional de reducció de soroll adient a la naturalesa i intensitat del soroll que s'ha generat.

En aquest estudi ens centrarem en els mètodes de filtratge local més utilitzats que interpreten el soroll com a un fenomen discret dins la matriu de mostreig. Alguns d'aquests són els següents:

· Filtres de mitjana i mediana

· Filtres gaussians

· Filtres bilaterals i conservatius

3. Descripció

Analitzarem pas a pas certs algorismes de reducció de soroll i veurem com aquests manipulen els valors del color dins els píxels de la imatge i entendrem fins a quin punt són capaços de complir amb el seu objectiu. També serà interessant analitzar quins efectes colaterals han generat tals processos sobre la qualitat de la imatge inicial. Què entén l'algorisme com a soroll? Es perd llum? Afecten també als píxels que no tenen soroll? Canvia la saturació de la fotografia? I el contrast? Quin algorisme és més eficient en cada cas?

Per poder desenvolupar un programa que permeti realitzar les tasques de generació gràfica s'utilitzarà Processing, ja que aquest entorn de desenvolupament multiplataforma és enfocat a projectes multimèdia d'aquest tipus i ja ha estat una matèria treballada en el grau precisament en la manipulació gràfica. Addicionalment, l'ús del llenguatge Java ens permetrà l'accés a diverses biblioteques externes quan sigui necessari.

En acabar l'estudi haurem obtingut suficients resultats com per poder generar, utilitzant el mètode científic, diverses afirmacions sobre els mateixos algorismes aplicats a les imatges processades. Aquests resultats totalment reproduïbles poden comunicar unes conclusions que intentarem desvetllar.

4. Objectius

Es tracta d'aportar cert coneixement que pot ser útil a l'hora de seleccionar un mètode de reducció de soroll, com a resultat d'un procés d'anàlisi metodològic tant sobre els algorismes utilitzats com els resultats obtinguts a partir d'aquests, donant una explicació dels avantatges o inconvenients que puguin sorgir en contrastar-los. Aquest estudi no només serà una resposta a les preguntes que es plantegen, sinó que s'intentarà justificar cada resultat obtingut, a partir de les seves causes sobre l'alteració dels píxels.

S'analitzaran diferents mecanismes independents del senyal que permeten reduir el soroll en les imatges digitals, veurem com afecten als píxels i al conjunt de la composició així com la seva eficiència i temps de processat. Avaluarem a la vegada l'efectivitat de cada algorisme definint certes variables en la interfície gràfica.

Amb l'objectiu de presentar l'estudi com una petita ajuda enfocada als fotògrafs i als editors d'imatges a l'hora de seleccionar un mètode de reducció

de soroll, aquest estudi serà adaptat per poder-se consultar en ordre invers, a partir dels resultats les corresponents explicacions i aquestes a la vegada a més baix nivell. També ha de presentar unes conclusions el més entenedores i esquemàtiques possibles per orientar aquests resultats a un públic més general.

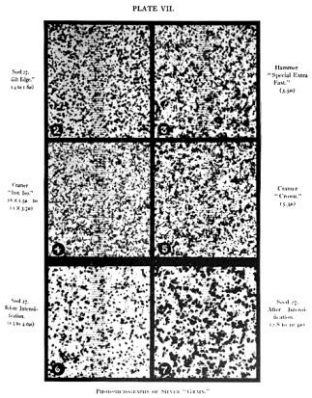
5. Marc teòric

5.1. Antecedents

Les imatges han estat cada vegada més presents en els mitjans de comunicació i aquestes quasi sempre es veuran afectades en certa mesura per el soroll, sobretot en fotografies amb una sensibilitat ISO alta. Així doncs, l'interès per rectificar aquestes pèrdues de qualitat també ha crescut molt últimament. Aquest problema no té una solució perfecta avui en dia, però en les darreres dècades s'han creat i millorat molts mètodes per reduir-lo, amb el propòsit de fer una estimació de la imatge original sense aquesta pertorbació del color, però evitant afectar-la. Amb tot, sovint és inevitable sacrificar detalls de la imatge tractada, contorns o línies fines durant aquest procés.

Sense anar més lluny, la sensibilitat ISO també la trobem en les fotografies analògiques: El film conté granets aleatoris d'halur de plata que s'enfosqueixen al ser exposats a la llum. Si aquests són grans i irregulars es pot captar la llum més ràpidament, però perdent definició i a la inversa.

Figura 5



WALLACE, Robert James. 1994. *Silver 'Grain' in Photography* [imatge]. Wikipedia. Disponible a: https://commons.wikimedia.org/wiki/File:Film_Grain.jpg

5.2. Escenari

Avui l'ús d'aquests algorismes no és exclusiu per a tècnics en disseny gràfic, sinó que les càmeres digitals ja les incorporen i es poden executar durant la mateixa presa, tal com passa també en els dispositius mòbils. És, podem dir, una part fonamental del procés de captura avui en dia i s'aplica fins i tot en temps real durant la gravació de vídeo.

Aquests mètodes es poden classificar segons la naturalesa de les tècniques utilitzades:

- Basats en mediana
    - Filtres mediana
    - Filtres mediana ponderada
    - Filtres mediana canviant
- Basats en models estadístics
    - Transformació amb ona discreta
    - Transformació amb ona complexa
    - Transformació de contorn discret
    - Transformació de corba discreta
- Basats en derivades
    - Difusió no lineal
    - Minimització funcional d'energia
- Basats en lògica difusa
    - Mètodes clàssics
    - Filtres de ponderació difusa
    - Filtres difuminats FIRE
- Basats en morfologia matemàtica
    - Filtres morfològics basats en elements d'estructura múltiple
    - Filtres morfològics suaus
    - Filtres morfològics basats en operadors híbrids

Actualment sembla que els algorismes que millors resultats donen són els basats en *deep learning*, un camp dins del *machine learning* (intel·ligència artificial) que utilitza xarxes neuronals per resoldre certs problemes complexos. L'inconvenient és que són aplicacions tancades i de pagament.

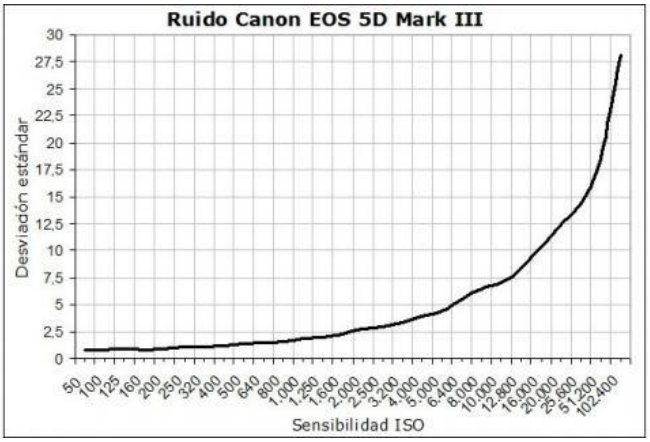
5.3. Bases teòriques de referència

5.3.1. Sensibilitat

Com ja hem comentat, la sensibilitat ISO en el moment de la captura és decisiu per prevenir la generació de soroll en la fotografia. Aquesta, representa el nivell

de sensibilitat a la llum permetent captar imatges en condicions de poca lluminositat, però també existeix soroll totalment inevitable i invariable en el temps com el FPN produït pels sensors digitals CMOS. Aquests sensors estan formats per milions de fotodíodes, que són diminuts elements sensibles a la llum.

Figura 6



GOMBAU, Carlos. 2013. *N/A* [imatge]. Computerhoy, Disponible a: https://computerhoy.com/paso-a-paso/fotografia/que-es-como-calcular-ruido-foto-photoshop-1982

5.3.2. Estimació del soroll

Si disposem d'una sola imatge, s'han proposat moltes tècniques per preveure el soroll previ a aplicar un algorisme de reducció de soroll, però solen ser adaptades a concrets problemes de soroll, per exemple l'originat durant els escanejos o analitzant l'espectre d'una font de soroll utilitzant certs llindars adaptatius per segmentar el gradient de la imatge (per localitzar zones més o menys suaus on aplicar-lo), sempre tenint en compte que la imatge conté àmplies zones homogènies i que el soroll és homogeni. D'altra manera, els resultats són sempre pobres.

5.3.3. Percepció

Dues universitats europees, l'Institut für Informatik VI d'Alemanya i l'School of Psychology del Regne Unit van realitzar un estudi experimental amb un grup de joves avaluant la percepció de la qualitat en imatges originals i manipulades amb soroll: "The analysis demonstrated that compared to the original high quality images, introducing noise has significantly reduced the perceived image quality ratings" (Röhrbein *et al.,* 2015)

5.4. Altres estudis similars

En el camí recorregut durant aquesta recerca he pogut observar que existeixen diversitat d'estudis més o menys semblants a aquest, sovint tractats des d'un punt de vista matemàtic, però pocs que expliquin clarament els efectes col·laterals de diversos algorismes en diferents aspectes gràfics de les fotografies.

Per exemple, podem tenir en compte els següents, també referenciats a la bibliografia:

- Review of Image Denoising Algorithms, with a New One
- A Study Of The Effects Of Gaussian Noise On Image Features
- Effect of Blur and Noise on Image Denoisingbased on PDE

6. Metodologia

6.1. Etapes

El projecte es planteja en una metodologia predictiva, ja que constarà de la successió de diferents etapes lineals, amb una gran presencia de planificació i recerca prèvies, però sent possible i no desitjable haver de treballar etapes anteriors durant el procés si fos necessari. L'objectiu idoni és progressar de forma lineal per obtenir els resultats complets en un temps definit:

- Recerca d'estudis similars en biblioteques, publicacions o estudis online. Referenciar-los.
    - Aprofundiment en aquest camp de coneixement i descobrir el funcionament de diferents algorismes, inclòs els experimentals.
    - Comprensió de com s'adapten els algorismes en Java. Cerca de biblioteques de suport.
    - Introducció de certs algorismes en l'estudi.
    - Definició de l'abast.
- Desenvolupament inicial en Processing.
    - Repàs de materials que expliquin el funcionament de Processing important i mostrant imatges.
    - Creació de diverses interfícies bàsiques en Processing que puguin mostrar dues imatges.
    - Desenvolupament de diferents algorismes en el paradigma POO i comprovar el correcte funcionament. Interpretar fórmules si és necessari en format RGB.
    - Establiment i representació de les constants o variables de cada algorisme en forma de controls. Cerca de biblioteques de suport.

- Possibilitar l'exportació de la imatge.
- Documentació del funcionament de cada classe creada i el funcionament dels algorismes sobre els píxels.

❖ Desenvolupament final en Processing.
- Unificació les diferents interfícies en una. Implementar un selector del mètode a aplicar. Desactivar o activar controls de les variables.
- (Opcional) Representació en forma de gràfiques de certes característiques de les imatges obtingudes i/o utilitzant matrius per analitzar-ne les diferències.
- *Testing*.
- Verificació de l'eficiència del codi.
- Representació del diagrama UML global.
- Maquetació global de la interfície.
- Generació d'executable.
- Anotació d'explicacions en les línies de codi.
- Creació d'un "*Readme*" i un manual d'usuari.

❖ Preparació d'un joc d'imatges de prova

❖ Anàlisi gràfic i presentació.
- Definició d'una metodologia de medició i representació dels resultats.
- Utilització de programari com Photoshop o ImageJ per analitzar i extreure dades de totes les imatges obtingudes i originals.
- Interpretació de les dades en informació, ponderant i és necessari, fent més objectius els resultats.
- Transformació dels resultats en format de conclusions obtingudes, aportant explicacions i conceptes breus si són necessaris.
- Presentació de gràfiques, taules, documents i elaboració de presentacions.

❖ Redacció de la memòria.

❖ Preparació de la presentació.

6.2. Recerca

El material que es necessita per poder arribar a conclusions objectives sobre aquests mètodes de reducció de soroll, són les dades extretes en questa etapa. Per tant, aquí s'haurà de posar èmfasi en extreure el màxim d'informació abans i després de cada procés, sempre amb la premissa bàsica de treballar sobre un mateix espai de color: l'RGB, evitant així treballar sobre dades esbiaixades. A vegades, pot semblar més útil utilitzar el model HSL per extraure valors de lluminositat o saturació, però significaria que els valors no es podrien comparar amb tots els altres i per precaució he preferit analitzar totes les dades a partir del model RGB.

Aquelles dades que no puguin ser obtingudes directament amb l'ús de programari extern (Photoshop, ImageJ...) s'hauran d'extreure utilitzant mètodes programats en Processing.

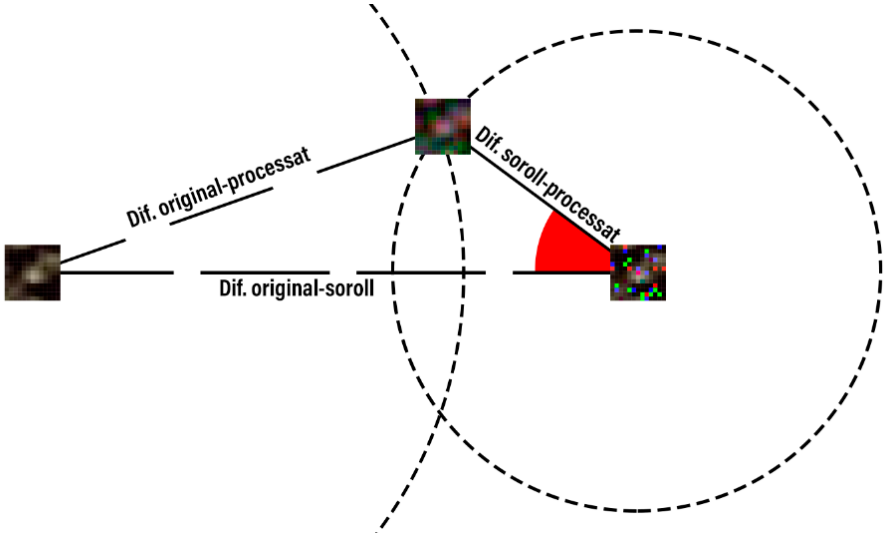
Així, per cada imatge individual -original sense soroll, original amb soroll i processada- i de forma relativa -processada + original i processada + original amb soroll-, podem obtenir una taula amb les següents característiques:

- Il·luminació global
    - Mitjana [Photoshop]
    - Mediana [Photoshop]
    - Desviació típica [Photoshop]
    - Moda [ImageJ]
    - Minim [ImageJ]
    - Màxim [ImageJ]
- Vermell global
    - Mitjana [Photoshop]
    - Mediana [Photoshop]
    - Desviació típica [Photoshop]
    - Moda [ImageJ]
    - Minim [ImageJ]
    - Màxim [ImageJ]
- Verd global
    - Mitjana [Photoshop]
    - Mediana [Photoshop]
    - Desviació típica [Photoshop]
    - Moda [ImageJ]
    - Minim [ImageJ]
    - Màxim [ImageJ]
- Blau global
    - Mitjana [Photoshop]
    - Mediana [Photoshop]
    - Desviació típica [Photoshop]
    - Moda [ImageJ]
    - Minim [ImageJ]
    - Màxim [ImageJ]
- Distribució de pesos
- Característiques gràfiques globals
    - Interpretació de la saturació [Processing]*
    - Valors extrems [Processing]*
    - Colors únics [Processing]*
    - Temps de processat [Processing]*
    - Diferències directes [Processing][1]

[1] Càlcul dels diferencials d'intensitat entre els píxels de les dues imatges situats en la mateixa posició.

* Al treballar amb Processing i posteriorment analitzar amb altres programes, necessitarem exportar necessàriament la imatge. D'entre els formats permesos pel *datatype* PImage (.gif, .jpg, .tga, i .png) utilitzarem sempre .png perquè no presenta pèrdues.

Durant l'anàlisi he considerat una altra característica subjectivament avaluable, la triangulació de les diferències entre imatge original, imatge amb soroll i imatge sense soroll. Si partim només dels valors soroll-original i processat-soroll, només sabrem si el procediment d'eliminació de soroll ha causat molta o poca variància respecte la imatge amb soroll, però no sabrem si ho ha fet per semblar-se més a la original. Per aconseguir-ho, podríem calcular una triangulació afegint el valor original-processat i calcular l'obertura de l'angle en el vèrtex amb soroll. El resultat serà favorable si aquest és menor.



Senzilla il·lustració del càlcul per la triangulació de les diferències entre imatges originals, amb soroll i processades.

Podríem simplement realitzar una comparació directa entre els valors original-processat i original-soroll per avaluar si l'aplicació de l'algorisme s'acosta al resultat desitjat, però això sovint estarà influenciat per la magnitud dels paràmetres. En canvi, si calculem l'angle, tindrem un indici de que l'algorisme "apunta" en la direcció correcta o en altres paraules, que independentment de la intensitat, aquest "ataca" la naturalesa del problema.

Una vegada obtinguts els valors, els compararem per paquets de original + soroll, d'aquesta manera obtindrem informació a partir d'aquestes dues constants, sobre la eficàcia dels 5 algorismes aplicats.

6.2.1. Mètodes

6.2.1.1. Mètodes d'eliminació de soroll

Els mètodes d'eliminació de soroll fruit de la primera recerca d'informació per internet, són els que, per la seva senzillesa, els he vist aptes per ser programats en Java i un per un els he interpretat des de la seva explicació i recolzant-me sobre els codis que presenten. En cap cas he fet una calca, primer perquè els llenguatges de programació són diferents a Java i segon perquè els he adaptat dins les possibilitats que disposa l'API de Processing. Alguns d'aquests algorismes, una vegada programats i aplicant certes variables, han servit també com si fossin algorismes diferents.

Mètode	Analitzat	Procedència	Paràmetres	Minim	Màxim
1 Gaussian Bilateral		Aliè	Mask	1	10
			Sigma	1	30
			Threshold	0	999
2 Crimmins Speckle Removal		Aliè	Step	1	25
			Margin	2	100
3.1 Non-Uniform Weight		Aliè	Mask	1	20
			Centralization	0	20
3.2 Uniform Weight		Aliè	Mask	1	20
			Centralization	0 *	
4 Bilateral		Aliè	Mask	1	10
			Threshold	0	999
			Colors	0	1
5 Gaussian		Aliè	Mask	1	5
			Sigma	10	30
6 Conservative		Aliè	Mask	1	20
7.1 Salt and Pepper Adapted Median		Aliè	Mask	1	20
			Threshold	0	255
			Colors	0	1
7.2 Median		Aliè	Mask	1	20
			Threshold	0 *	
			Colors	0	1
8 Average		Aliè	Mask	1	20
9 Progressive Median		Propi	Initial Mask Y	1	99
			Initial Mask X	1	99
			Mask	1	20
			Threshold	1	255
			Colors	0	1
10 Progressive Conservative		Aliè	Initial Mask Y	1	99
			Initial Mask X	1	99
			Mask	1	20
			Threshold	1	255
			Colors	0	1
11 Extreme Median		Propi	Mask	1	20
			Threshold	0	100

6.2.1.1.1 Gaussian Bilateral

- Explicació: Es tracta d'utilitzar a la vegada una funció gaussiana, que definirà el pes que cada píxel de la màscara tindrà sobre el píxel central, juntament amb la detecció de contorns que ofereix el filtre bilateral, que informarà dels píxels que s'han de tenir en

compte o no segons la variació lumínica entre aquests (perifèrics respecte el central), semblant al Crimmis Speckle Removal (algorisme 2). Sempre que s'apliqui una reducció de soroll es durà a terme un càlcul de mitjana ponderada en els pesos en la distribució gaussiana.

- Concepte de soroll: Desconeixem per complet on pot haver presència de soroll, però sabem que els sensors han rebut una desviació de la llum captada i aquesta segurament es troba en els píxels més propers i menys en els llunyans, seguint així una distribució gaussiana. Ja que prop dels píxels també es poden donar salts naturals d'intensitat en la imatge original (contorns), descartem del càlcul aquells que representen un salt d'intensitat important, evitant així la contaminació de la mitjana i preservant els contorns, a canvi de no poder detectar tant bé el soroll prop d'aquests salts.
- Paràmetres
    - Mask: Mida de la màscara. Degut a que les cares d'una màscara no poden ser d'un nombre parell d'elements ni ser 1, la mida real de la màscara és el resultat de calcular (Mask*2)+1
    - Sigma: Aquest valor es fa servir, entre altres, per al càlcul de les desviacions estàndard i es pot representar com σ. Representa la dispersió, des del centre de la màscara, on estimem que s'han perdut les dades cromàtiques del píxel. És a dir, en un valor alt representa una campana més ample i en el càlcul es donarà una importància semblant tant als píxels propers com els més llunyans.
    - Threshold: Durant la detecció de contorns, aquest llindar representa quant de restrictiu és l'algorisme respecte tenir en compte o no valors amb salts d'intensitat entre ells. A més valor, més es respecten els contorns.

6.2.1.1.2 Crimmins Speckle Removal

- Explicació: Aquest algorisme està dissenyat per eliminar el soroll de sal i pebre i és més efectiu quantes més iteracions s'apliquin, però amb l'inconvenient de perdre definició. El càlcul per detectar el soroll es fa a partir d'avaluar les diferències d'intensitat entre els píxels oposats respecte el central. En una màscara de 3x3, el càlcul mesura les diferències d'intensitat en les 8 direccions (N,N-E,E,S-E,S,S-O,O,N-O) i en els dos sentits, detectant diferències que superin el llindar establert i corregint si es dóna el cas.
- Concepte de soroll: Desconeixem per complet on pot haver presència de soroll, però sospitem que sol estar en píxels que presenten salts d'intensitat entre els que té al voltant.
- Paràmetres

- Step: En aplicar les correccions, aquest paràmetre representa la duresa d'aquestes. A més valor, més variacions d'intensitat aplicarem.
- Margin: En avaluar les diferències d'intensitat, apliquem un marge de tolerància. A més valor, més permissiu és l'algorisme.

6.2.1.1.3 Non-Uniform Weight

- Explicació: Es tracta d'un algorisme simple en el que es dóna un pes ponderat als píxels circumdants segons la distància d'aquests respecte del píxel central. Per definició s'utilitza una fórmula senzilla amb una variable, per poder manipular el nivell de concentració dels pesos en el centre de la màscara. Fórmula aplicada = $1/((n \cdot d)+1)$, on n és la variable i d és la distància (x+y) respecte el píxel central. D'aquesta manera, els pesos es tornen variables sobretot per la distància però també segons el factor que li introduïm. Si ho configurem amb n = 0 el resultat és una màscara plana amb un pes equivalent en tota la màscara, és a dir, ens serveix també com a màscara de pes uniforme (Uniform Weight).
- Concepte de soroll: Desconeixem per complet on pot haver presència de soroll, però pensem que ens podem acostar a la informació original de cada píxel analitzat tota la màscara, especialment prop del centre.
- Paràmetres
  - Mask: Mida de la màscara. Com que les cares d'una màscara no poden ser d'un nombre parell d'elements ni ser 1, la mida real de la màscara és el resultat de calcular (Mask*2)+1
  - Centralization: Segons l'explicació, representa la variable n. Un valor alt representa que s'atorga més pes als valors centrals i menys als llunyans.

6.2.1.1.4 Bilateral

- Explicació: És un algorisme simple que detecta els contorns de forma bilateral, informant dels píxels que s'han de tenir en compte o no, segons la variació lumínica entre aquests (perifèrics respecte el central). La variable llindar servirà per marcar un marge de tolerància, a la resta de píxels se'ls atribueix un pes diferent segons el grau de variació lumínica.
- Concepte de soroll: Desconeixem per complet on pot haver presència de soroll, d'aquesta manera tenim en compte tots els píxels de la màscara de la mateixa manera. Ja que dins la màscara també es poden donar salts naturals d'intensitat en la

imatge original (contorns), descartem del càlcul aquells que representen un salt d'intensitat important, evitant així la contaminació de la mitjana i preservant els contorns, a canvi de no poder detectar tant bé el soroll prop d'aquests salts.

- Paràmetres
    - Mask: Mida de la màscara. Degut a que les cares d'una màscara no poden ser d'un nombre parell d'elements ni ser 1, la mida real de la màscara és el resultat de calcular (Mask*2)+1.
    - Threshold: Durant la detecció de contorns, aquest llindar representa quant de restrictiu és l'algorisme respecte tenir en compte o no valors amb salts d'intensitat entre ells. A més valor, més es respecten els contorns.
    - Colors: En el moment d'aplicar el llindar per decidir si un píxel es té en compte o no per al càlcul, es poden tenir en compte els tres canals de color per separat o bé conjuntament: 0 significa conjuntament. Llavors, si qualsevol dels 3 canals sobrepassa el llindar, el píxel no es tindrà en compte i per tant el filtre serà menys agressiu davant canvis d'intensitat i evitarà canvis en el color resultant.

6.2.1.1.5 Gaussian

- Explicació: És un algorisme que es basa únicament en la distribució dels pesos dins la màscara seguint una distribució en forma gaussiana en dues dimensions. Aquests pesos podran variar depenent de la mida de la màscara i del factor sigma.
- Concepte de soroll: Interpretem que la imatge ha patit una desviació de la llum captada en tots els píxels, equitativament. Aquesta desviació pensem que, per un principi de probabilitat, trobarem aquestes desviacions en més quantitat en els píxels propers que no pas en els llunyans. Ja que ha estat provocat per una causa natural de la llum, aquesta probabilitat prendrà una forma gaussiana.
- Paràmetres
    - Mask: Mida de la màscara. Degut a que les cares d'una màscara no poden ser d'un nombre parell d'elements ni ser 1, la mida real de la màscara és el resultat de calcular (Mask*2)+1.
    - Sigma: Aquest valor es fa servir entre altres en el càlcul de les desviacions estàndard i es pot representar com σ. Representa la dispersió, des del centre de la màscara, on estimem que s'han perdut les dades cromàtiques del píxel. És a dir, en un valor alt representa una campana més ample i en el càlcul es donarà tanta importància als píxels propers com els més llunyans.

6.2.1.1.6 Conservative

- Explicació: Aquest algorisme corregeix les intensitats dels píxels només si aquests superen les intensitats màximes i mínimes dels que l'envolten. En tal cas, el píxel s'ajusta a aquestes.
- Concepte de soroll: Entenem que el soroll que afecta la imatge consta de desviacions exagerades d'intensitat causant valors extrems i poc comuns en la imatge, però localitzat en píxels aleatoris i dispersos entre ells, semblant a un soroll impulsional poc freqüent.
- Paràmetres
    - Mask: Mida de la màscara. Degut a que les cares d'una màscara no poden ser d'un nombre parell d'elements ni ser 1, la mida real de la màscara és el resultat de calcular (Mask*2)+1.
    - Colors: Discerneix sobre si l'algorisme ha de tractar els canals de color per separat o conjuntament. 1 = per separat.

6.2.1.1.7 Salt and Pepper Median

- Explicació: Per cada píxel de la imatge, l'algorisme calcula la mediana del seu entorn i compara els valors obtinguts, amb el píxel central. Si la diferència és gran, es substituirà el valor del píxel central per la mediana obtinguda. S'aplica un marge de tolerància com a variable i de ser 0, l'algorisme operaria com un algorisme de mediana pur.
- Concepte de soroll: El soroll causa en la imatge píxels aleatoris amb valors fora del comú. Els píxels afectats poden estar agrupats però sempre tindran un valor diferent de l'habitual en el seu voltant.
- Paràmetres
    - Mask: Mida de la màscara. Degut a que les cares d'una màscara no poden ser d'un nombre parell d'elements ni ser 1, la mida real de la màscara és el resultat de calcular (Mask*2)+1.
    - Threshold: Marca la tolerància entre valors d'un píxel i la mediana de la màscara. Valor alt = més tolerància a les diferències.
    - Colors: Definirà si tractem els tres colors conjuntament o per separat (0 = conjuntament).

6.2.1.1.8 Average

- Explicació: La màscara que convoluciona tota la imatge calcula una mitjana aritmètica d'intensitats de color i el resultat s'aplica al píxel central tractat en cada cas.
- Concepte de soroll: Desconeixem per complet on se situa el soroll i quina intensitat té, però creiem que pot estar repartit per tota la imatge amb una probabilitat desconeguda però constant. El soroll no és agressiu i no causa valors estranys ni extrems.
- Paràmetres
    - Mask: Mida de la màscara. Degut a que les cares d'una màscara no poden ser d'un nombre parell d'elements ni ser 1, la mida real de la màscara és el resultat de calcular (Mask*2)+1.

6.2.1.1.9 Progressive Median

- Explicació: Aquest algorisme es recolza també sobre els píxels que ja ha tractat, disminuint la contaminació del soroll en els càlculs posteriors però augmentant el risc de propagar errors. Comença en una posició seleccionada i calcula la mediana. A partir d'aquesta posició, la convolució es duu a terme de forma circular fins el punt de tractar tots els píxels de la imatge. Durant cada iteració aplica una mediana amb preferència per incloure en el càlcul els píxels ja tractats.
- Concepte de soroll: El concepte de soroll és el mateix cas que la mediana, el soroll causa en la imatge píxels aleatoris amb valors fora del comú. Els píxels afectats poden estar agrupats però sempre tindran un valor diferent de l'habitual en el seu voltant.
- Paràmetres
    - Initial Mask X: Posició aproximada de l'eix X de la primera aplicació de la màscara. Es fan servir valors de l'1 al 99.
    - Initial Mask Y: Posició aproximada de l'eix Y de la primera aplicació de la màscara. Es fan servir valors de l'1 al 99.
    - Mask: Mida de la màscara. Degut a que les cares d'una màscara no poden ser d'un nombre parell d'elements ni ser 1, la mida real de la màscara és el resultat de calcular (Mask*2)+1.
    - Threshold: Marca la tolerància entre valors d'un píxel i la mediana de la màscara. Valor alt = més tolerància a les diferències.
    - Colors: Definirà si tractem els tres colors conjuntament o per separat (0 = conjuntament).

6.2.1.1.10 Progressive Conservative

- Explicació: Aquest algorisme es recolza també sobre els píxels que ja ha tractat, disminuint la contaminació del soroll en els càlculs posteriors però augmentant el risc de propagar errors. Comença en una posició seleccionada i la processa amb el mètode Conservative. A partir d'aquesta posició, la convolució es duu a terme de forma circular fins el punt de tractar tots els píxels de la imatge. En cada iteració, s'aplica un altre cop el mètode conservatiu amb preferència per incloure en el càlcul els píxels ja tractats.
- Concepte de soroll: el concepte de soroll és el mateix cas que el mètode Conservative, entenem que el soroll que afecta la imatge consta de desviacions exagerades d'intensitat causant valors extrems i poc comuns en la imatge, però localitzat en píxels aleatoris i dispersos entre ells, semblant a un soroll impulsional poc freqüent.
- Paràmetres
    - Initial Mask X: Posició aproximada de l'eix X de la primera aplicació de la màscara. Es fan servir valors de l'1 al 99.
    - Initial Mask Y: Posició aproximada de l'eix Y de la primera aplicació de la màscara. Es fan servir valors de l'1 al 99.
    - Mask: mida de la màscara. Degut a que les cares d'una màscara no poden ser d'un nombre parell d'elements ni ser 1, la mida real de la màscara és el resultat de calcular (Mask*2)+1.
    - Threshold: Marca la tolerància entre valors d'un píxel i la mediana de la màscara. Valor alt = més tolerància a les diferències.
    - Colors: Definirà si tractem els tres colors conjuntament o per separat (0 = conjuntament).

6.2.1.1.11 Extreme Median

- Explicació: Senzill algorisme que analitza la imatge en busca de píxels amb valors extrems. Quan en troba un, hi aplica el mètode de la mediana.
- Concepte de soroll: La imatge està afectada per un soroll amb distribució aleatoria que causa valors extrems en els tres canals a la vegada, obtenint blancs o negres purs o quasi purs.
- Paràmetres
    - Mask: Mida de la màscara. Degut a que les cares d'una màscara no poden ser d'un nombre parell d'elements ni ser 1, la mida real de la màscara és el resultat de calcular (Mask*2)+1.

- Threshold: Defineix el concepte de valor extrem. Quant més alt sigui aquest paràmetre, més rang d'intensitats a partir dels extrems es comptaran com a valors extrems.

Els mètodes que he seleccionat per realitzar l'anàlisi de dades han estat els següents. No he tingut un criteri objectiu, sinó que he seleccionat els que crec que són més senzills, que més s'utilitzen i més s'enfoquen a una sola característica, evitant els algorismes compostos com ara Conservative Progressive Median. D'aquesta manera, podrem arribar a conclusions més enfocades a cada mètode, podent fer-se una idea després dels avantatges d'altres mètodes que són compostos d'aquests més simples.

- Average
- Gaussian
- Bilateral
- Salt and Pepper Median
- Conservative

6.2.1.2. Mètodes d'addició de soroll

Per poder treure conclusions adaptades a cada naturalesa de soroll, la millor opció és generar el soroll que després eliminarem, de manera idèntica. Per això, a partir de certes imatges sense soroll d'entrada, he utilitzat únicament una eina web que es diu Mass-images.pro. D'aquesta manera, podem obtenir per separat diverses naturaleses de soroll aplicades en la mateixa mesura a diferents imatges.

Tipus de soroll aplicats a les imatges originals:

· Impulse: Podria haver-se generat degut a errors de captura o transmissió de l'aparell.

· Gaussian: Podria haver-se generat per calor en els sensors o transmissions defectuoses.

· Poisson: Significa soroll de captura. Prové de la naturalesa de la llum, que genera fluctuacions entre els fotons capturats pel sensor.

6.2.2. Imatges Originals

Les imatges originals les he descarregat de la pàgina web pexels.com perquè permet descarregar fotografies d'alta qualitat amb un ús gratuït i lliures de llicència. L'objectiu ha estat trobar imatges amb certes característiques que s'adaptin a l'ús que volem donar:

· Relativament fosques: Normalment el soroll és additiu, per tant serà més fàcil d'apreciar-lo en imatges fosques. També visualment es podrà percebre millor les diferencies de llum entre píxels foscos que entre píxels il·luminats.

· Mida FHD: Totes les imatges han de ser exactament de 1920 x 1080 píxels, sent necessari tallar-les totes a aquesta mida estàndard.

· Detall: El contingut de les imatges ha de tenir certa qualitat en quant a definició. No ens interessen fotografies borroses on sigui més difícil percebre el soroll. També necessitarem comprovar si els algorismes respecten els contorns.

· Sense soroll: Les imatges d'origen aparentment no han de ser afectades per cap tipus de soroll.

6.3. Anàlisi i conclusions

6.3.1 Metodologia

Encara que, des de la perspectiva creativa, aquests anàlisis no defineixin la qualitat de les fotografies, si podrem interpretar les variacions entre els valors anteriors i posteriors a cada processat. Una vegada haguem recopilat tota la informació s'hauran de generar unes altres taules amb les informacions relatives, establint com a rang de representació aquells màxims i mínims obtinguts en tot l'estudi, de manera que es puguin representar les dades en gràfics i percebre millor les diferències. Finalment i a partir d'aquestes, aplicarem uns barems -aquesta dada ha de ser necessàriament més subjectiva- que permeti extreure informació útil sobre si els algorismes són més o menys invasius i fins a quin punt són capaços de complir amb els seus objectius.

En aquest estudi no podem ser tan exhaustius com per abastar tots els escenaris possibles. És a dir, per treure conclusions de pes i ben fonamentades, hauríem de tenir en compte l'anàlisi en totes aquestes dimensions a la vegada: Diferents tipus d'imatges (blanc i negre, escala de grisos, color...), diferents naturaleses de soroll (impulsional, de fons, cromàtic...), diferents intensitats de soroll i fins i tot diferents combinacions en les variables dels algorismes de reducció. Per tot això, s'han de reduir aquestes possibilitats amb la finalitat d'equilibrar la balança d'objectivitat sobre abast realitzable.

Per prendre aquesta decisió fonamental en aquest treball, abans s'ha de tenir en compte el context actual i saber què s'espera obtenir, és a dir, quina serà la utilitat de les conclusions finals. Aquest treball no té l'objectiu d'informar sobre com aplicar un cert algorisme per obtenir millors resultats, per tant la dimensió de les variables en els algorismes no és tan rellevant. En canvi, si que ho són l'aplicació de tots els algorismes possibles i en certa mesura rellevant el grau de soroll de les imatges inicials. En conclusió, podrem ometre l'estudi sobre les variables i enfocar-nos més sobre la diversitat d'algorismes aplicats a la diversitat d'imatges i tipus de soroll. Tenim en compte $n \cdot x \cdot y \cdot z = 3 \cdot 3 \cdot 5 \cdot 2 = 90$ anàlisis on n són les imatges originals, x són les naturaleses del soroll, y són els diferents algorismes aplicats i z les intensitats dels seus paràmetres, que han de ser discretes però amb un efecte visible, per no generar resultants massa artificials que puguin donar després valors dispars.

Una vegada obtingudes totes les dades generades, tant en l'observació de les imatges com dels processos, ens farem certes preguntes que puguin ser respostes amb les dades obtingudes. Si més no, obtenir uns indicadors, una dada subjectiva que posi de manifest les diferències entre els algorismes. Cada algorisme haurà de respondre a les preguntes plantejades tenint en compte un conjunt relativament heterogeni de resultats (aplicat en diferents imatges, sorolls, intensitats...). D'aquesta manera, obtindrem certa confiança en aquests indicadors, suficient com per poder treure després algunes conclusions. En quant a l'anàlisi basat en les diferències entre imatges, s'estableix una quantificació relativa al valor original-soroll. A partir d'aquests percentatges, podrem validar les hipòtesis que es proposen al final del treball.

Finalment, partint de les respostes, podrem construir una o diverses taules proposant els millors o pitjors algorismes, davant certs problemes, així com els efectes colaterals d'aquestes aplicacions en altres característiques de la imatge.

6.3.2. Valoracions

Es representaran una sèrie de taules que emmagatzemaran les valoracions de certs aspectes sobre certes característiques de les imatges, sobre la seva recuperació o conservació originals. Per això, cada una s'adapta a rangs de valors del 0 al 10, a vegades assignats subjectivament a partir fruit de l'observació, sempre i quant no es pugui obtenir una informació basada en altres valors numèrics. En els casos en que es puguin obtenir de forma més objectiva es donarà una breu explicació a peu de taula sobre les operacions dutes a terme per arribar a cada resultat, amb l'objectiu de mantenir una fiabilitat en els resultats així com per permetre la seva reproducció per altres observadors.

Una vegada obtinguts tots els valors, es ponderaran segons l'objectivitat amb la que s'han obtingut, reflectint la confiança que tenim en aquella dada. Després, totes elles es mesclaran per obtenir una puntuació per cada parella soroll-mètode, ponderant una altra vegada sobre un valor de rellevància que representarà la importància que es podria atorgar a cada valor en quant a la qualitat d'una imatge. No es pot atorgar el mateix pes a cada característica i tampoc son valors determinats en el món de l'art així que la última solució decidir-los subjectivament.

6.4. Investigació

A l'hora de proposar unes hipòtesis he intentat trobar un punt mig entre ser específic, cosa que aportaria veracitat i objectivitat en la demostració, i genèric, que permetria abastar conceptes més amplis i per tant obtindria més sentit pràctic. Per això, m'he centrat en fer una hipòtesi sobre cada tipus de soroll i intentar demostrar si és possible la seva restauració, sense entrar en els mètodes o en les variables. En aquestes hipòtesis, parlem de restaurar i no pas de reduir el soroll, ja que compararem els valors original-processat sobre original-soroll i per tant no aprofitarem la direccionalitat teòrica de l'aresta soroll-processat, sinó que observarem directament si la imatge resultant s'assembla més a la original.

El valor de tall que marquin les hipòtesis, per precaució tindran aplicada una reducció de com a mínim el 10% al valor utilitzat per refutar-lo, tenint en compte que hi ha certa incertesa en quant al soroll introduït a les imatges, els gràfics originals o l'espectre de colors, per exemple.

7. Planificació

Una vegada plasmades les tasques amb les seves corresponents dependències, simultaneïtats, fites i durades estimades, s'han aplicat les desviacions de temps i adaptat al calendari del semestre coincidint a temps per les entregues de les PACs. Es partirà d'aquest cronograma, tot i que es portarà un control i es pot adaptar a canvis imprevistos:

- Recerca d'estudis similars en biblioteques, publicacions o estudis online. Referenciar-los.
    - Aprofundiment en aquest camp de coneixement i conèixer el funcionament de diferents algorismes, inclòs els experimentals.

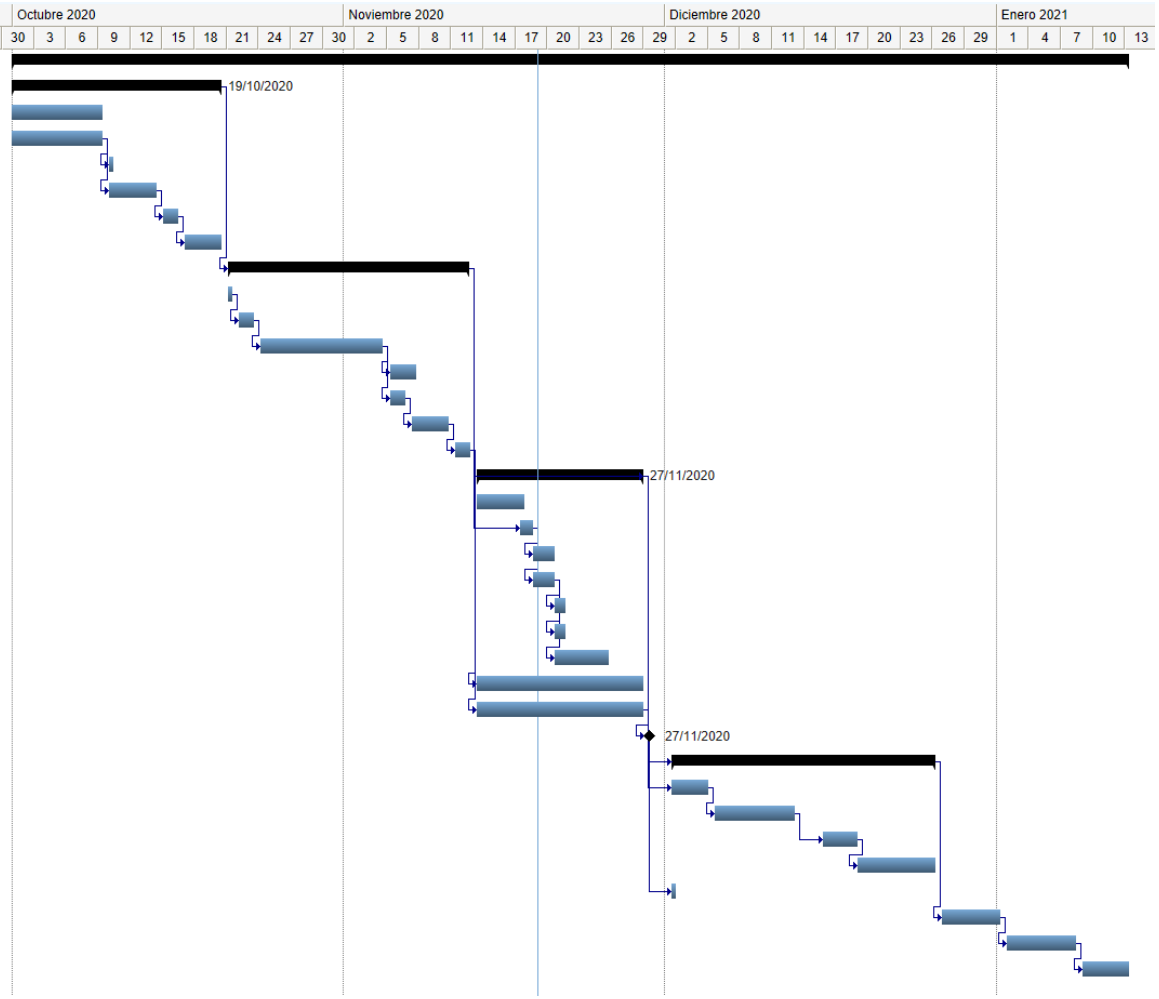
    - Comprensió de com s'adapten els algorismes en Java. Cerca de biblioteques de suport.
    - Introducció de certs algorismes en l'estudi.
    - Definició de l'abast.
    - Redacció de la memòria.
    - Entrega PAC2.
- Desenvolupament inicial en Processing
    - Repàs de materials que expliquin el funcionament de Processing important i mostrant imatges.
    - Creació de diverses interfícies bàsiques en Processing que puguin mostrar dues imatges.
    - Desenvolupament de diferents algorismes en el paradigma POO i comprovar el correcte funcionament. Interpretar fórmules si és necessari en format RGB.
    - Establir i representar les constants o variables de cada algorisme en forma de controls. Cerca de biblioteques de suport.
    - Possibilitar l'exportació.
    - Documentació del funcionament de cada classe creada i el funcionament dels algorismes sobre els píxels.
    - Redacció de la memòria.
- Desenvolupament final en Processing
    - Unificació les diferents interfícies en una. Implementar un selector del mètode a aplicar. Desactivar o activar controls de les variables
    - Testing.
    - Preparació d'un joc d'imatges de prova.
    - Representació en Processing de gràfiques de certes característiques de les imatges obtingudes i/o utilitzant matrius per analitzar-ne les diferències.
    - Anotació d'explicacions en les línies de codi.
    - Creació d'un Readme i un manual d'usuari.
    - Utilització de programari com Photoshop o ImageJ per analitzar i extreure dades de totes les imatges obtingudes i originals.
    - Redacció de la memòria.
    - Redacció del TFG.
    - Entrega PAC3.
- Anàlisi gràfic i presentació
    - Definició d'una metodologia d'anàlisi i de representació dels resultats.
    - Interpretació de les dades en informació, ponderant i és necessari, fent més objectius els resultats.
    - Transformació dels resultats en format de conclusions obtingudes, portant explicacions i conceptes breus si són necessaris.
    - Presentació de gràfiques, taules, documents i elaboració de presentacions.
    - Generació d'executable.
- Redacció final de la memòria i el TFG
- Preparació de la presentació
- Entrega final

Accés al PDF del diagrama de Gantt:

drive.google.com/file/d/15G5W4T6i7cz7xQ6RiK6i7pkhQFHI0x40/view?usp=sharing

Vista prèvia del diagrama de Gantt:

	EDT	Nombre	Duración	Inicio	Fin	Predecesoras
1	**1**	**⊟TFG**	**73.75días?**	**30/09/2020**	**11/01/2021**	
2	**1.1**	**⊟ Recerca d'estudis similars en biblioteques, publicacions o estuc**	**14días**	**30/09/2020**	**19/10/2020**	
3	1.1.1	Aprofundiment en aquest camp de coneixement i conèixer el fun	7días	30/09/2020	08/10/2020	
4	1.1.2	Comprenció de com s'adapten els algorismes en Java. Cerca de	7días	30/09/2020	08/10/2020	
5	1.1.3	Introducció de certs algorismes en l'estudi.	1día	09/10/2020	09/10/2020	4
6	1.1.4	Definició de l'abast	3días	09/10/2020	13/10/2020	4
7	1.1.5	Redacció de la memòria	2días	14/10/2020	15/10/2020	6
8	1.1.6	Entrega PAC2	2días	16/10/2020	19/10/2020	7
9	**1.2**	**⊟Desenvolupament inicial en Processing**	**17días**	**20/10/2020**	**11/11/2020**	**2**
10	1.2.1	Repàs de materials que expliquin el funcionament de Processing	1día	20/10/2020	20/10/2020	
11	1.2.2	Creació de diverses interfícies bàsiques en Processing que pugu	2días	21/10/2020	22/10/2020	10
12	1.2.3	Desenvolupament de diferents algorismes en el paradigma POO	8días	23/10/2020	03/11/2020	11
13	1.2.4	Establir i representar les constants o variables de cada algoriseme	3días	04/11/2020	06/11/2020	12
14	1.2.5	Possibilitar l'exportació	2días	04/11/2020	05/11/2020	12
15	1.2.6	Documentació del funcionament de cada classe creada i el funci	2días	06/11/2020	09/11/2020	14
16	1.2.7	Redacció de la memòria	2días	10/11/2020	11/11/2020	15
17	**1.3**	**⊟Desenvolupament final en Processing**	**12días**	**12/11/2020**	**27/11/2020**	**9**
18	1.3.1	Unificació les diferents interfícies en una. Implementar un selecto	3días	12/11/2020	16/11/2020	
19	1.3.2	Testing	1.5días	16/11/2020	17/11/2020	9
20	1.3.3	Preparació d'un joc d'imatges de prova	2días	17/11/2020	19/11/2020	19
21	1.3.4	Representació en Processing de gràfiques de certes característic	2días	17/11/2020	19/11/2020	19
22	1.3.5	Anotació d'explicacions en les línies de codi	1día	19/11/2020	20/11/2020	21
23	1.3.6	Creació d'un Readme i un manual d'usuari	1día	19/11/2020	20/11/2020	21
24	1.3.7	Utilització de programari com Photoshop o ImageJ per analitzar i	3días	19/11/2020	24/11/2020	21
25	1.3.8	Redacció de la memòria	12días	12/11/2020	27/11/2020	16
26	1.3.9	Redacció del TFG	12días	12/11/2020	27/11/2020	16
27	1.3.10	Entrga PAC3	0día	27/11/2020	27/11/2020	26
28	**1.4**	**⊟Anàlisi gràfic i presentació**	**19días?**	**30/11/2020**	**24/12/2020**	**17**
29	1.4.1	Definició d'una metodologia d'anàlisi i de representació dels resu	4días	30/11/2020	03/12/2020	17
30	1.4.2	Interpretació de les dades en informació, ponderant i és necessa	6días	04/12/2020	11/12/2020	29
31	1.4.3	Transformació dels resultats en format de conclusions obtingude	3.5días	14/12/2020	17/12/2020	30
32	1.4.4	Presentació de gràfiques, taules, documents i elaboració de pres	5.5días	17/12/2020	24/12/2020	31
33	1.4.5	Generació d'executable	1día?	30/11/2020	30/11/2020	27
34	1.5	Redacció final de la memòria i el TFG	4días	25/12/2020	30/12/2020	28
35	1.6	Preparació de la presentació	5días	31/12/2020	06/01/2021	34
36	1.7	Entrega final	2.75días	07/01/2021	11/01/2021	35

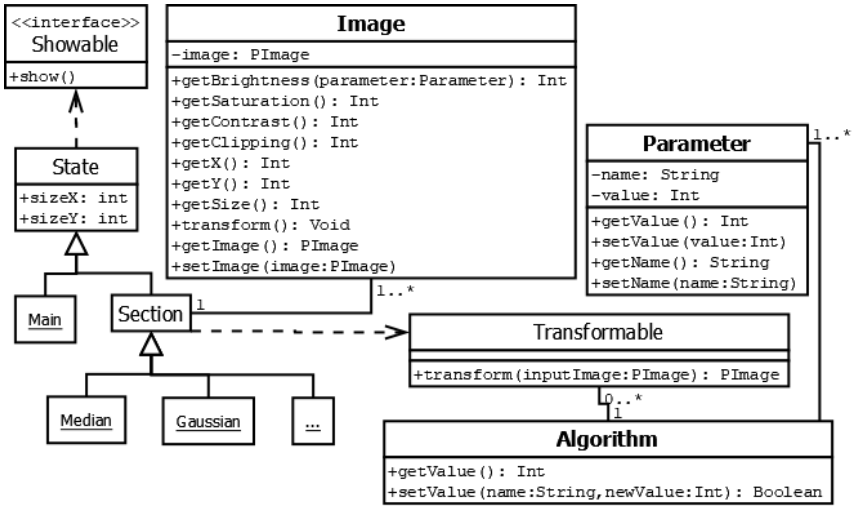


8. Arquitectura de l'aplicació

Com arrel del programa tindrem unes instàncies ja creades i configurades de la classe State, com poden ser la pantalla principal Main, l'lalgorisme Median, etc. Tots ells han de poder ser intercanviables fàcilment, per això existeix la dependència cap a la interfície Showable. Totes les vistes estaran configurades en quant a la distribució i representació dels elements gràfics en la finestra. Totes excepte Main també estaran obligades a definir un sol algorisme per transformar les imatges, el qual ha de contenir diversos paràmetres depenent de quin algorisme generem, per exemple: "dimensions de la mascara = 3". Ja que cada vista recopilarà i mostrarà uns paràmetres diferents i en posicions diferents, aquests seran variables en quantitat, valor i posició. De la mateixa manera, cada vista també podrà contenir imatges, com a mínim la d'entrada, que les representarà de manera particular.

Els algorismes hauran de contenir els paràmetres, que llegiran just abans de cada execució. També hauran de gestionar les peticions de l'usuari en el moment de modificar qualsevol valor.

La classe Imatge conté de forma definida els mètodes bàsics aplicables a qualsevol imatge, i com sempre és la classe State que ha de definir on representar la imatge rebuda.



Restriccions d'integritat:

- Els valors dels paràmetres no poden ser arbitraris, han d'estar dins uns valors màxims i mínims.
- Els paràmetres han de coincidir amb els mateixos que la Imatge de la vista.

Segurament es modifiquin certs aspectes o s'hagin obviat d'altres, per exemple la càrrega d'imatges des de les vistes és possible que no es pugui realitzar, o que duent a terme la implementació en codi sorgeixin necessitats i limitacions noves.

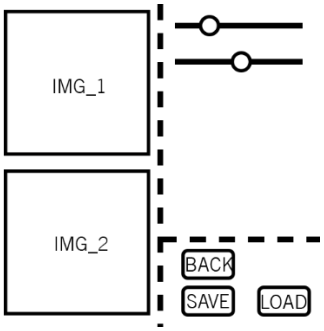
9. Plataforma de desenvolupament

El desenvolupament de l'aplicació d'anàlisi es realitzarà sobre Processing, un entorn de desenvolupament de codi obert que treballa sobre Java i està orientat sobretot a projectes d'ensenyança en matèria gràfica.

Em sembla molt adequat aquest entorn per dissenyar una aplicació senzilla com aquesta, ja que permet operar i representar dades gràfiques a través de la seva API d'una forma molt directa i amb poques línies de codi. La interfície gràfica d'usuari és molt intuïtiva permetent executar amb un clic així com l'execució en un sol clic, però sobretot perquè al compilar en Java permet també programar en Java pur si ho necessitem, podent així optar a gran varietat de llibreries si fos necessari.

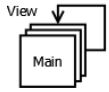
10. Prototips

Els elements visuals de cada vista existiran d'acord amb les variables que l'usuari pot modificar depenent de l'algorisme seleccionat. D'aquesta manera, cada *wireframe* pot ser molt diferent d'un altre, però per norma general haurien de respectar una mateixa disposició dels elements: Les imatges una sobre l'altra ocupant com a màxim tota l'alçada (re-escalant si és necessari) i en la meitat dreta una successió d'elements que permetin introduir valors. En l'extrem dret inferior, un botó de retrocés, un de càrrega d'imatge i un de guardat.



La navegació serà molt simple, en l'inici de l'aplicació s'executarà en la vista principal que contindrà un accés (botó) per a cada algorisme i en aquests un de retrocés al principal. D'aquesta manera es poden implementar els algorismes independentment per després unificar-los com a vistes en el projecte principal. D'aquesta manera, es facilita el procés de *testing* dels diferents algorismes, ja

que podem definir imatges per defecte en la carpeta de recursos i també per la incertesa inicial en el mateix èxit de la implementació d'un algorisme.



Els processos de *testing* es duran en el mateix temps de la programació, però es posarà especial èmfasi en la corresponent fase prevista en el diagrama de Gantt, prèviament a l'entrega del producte finalitzat, d'acord amb el mètode predictiu del projecte. Seria avaluable un risc en aquest aspecte, pot passar que en aquesta fase sorgeixin molts errors o *bugs* que corregir i faci endarrerir l'entrega, però amb les característiques d'aquest projecte: Petit, abast molt ajustat i amb terminis molt curts, seria difícil que el risc sigui massa alt.

11. Perfils d'usuari

L'aplicació està oberta a tot tipus d'usuaris. El propòsit en la creació d'aquest programa és que serveixi com una eina més d'anàlisi d'imatges que faciliti l'experimentació, però enfocada en certs algorismes de reducció de soroll, problema comú tant en usuaris novells, experts com professionals. D'aquesta manera, no existeix un perfil definit més enllà de qualsevol en el món de la creació gràfica.

Pel que fa al treball, la matèria que s'hi desenvolupa sol requerir de coneixements en el camp de la informàtica i del disseny gràfic. No obstant, serà adaptada en certa mesura a permetre l'accés al coneixement des d'un perfil més general a través de referències o explicacions.

Bibliografia

Il·lustracions:

1. MDF, 2010. Photon-Noise.Jpg. [imatge] Disponible a: <https://upload.wikimedia.org/wikipedia/commons/4/42/Photon-noise.jpg> [Accedit 28 Setembre 2020]. (CC BY-SA 3.0)
2. Lorenzi, L., 2010. Salt And Pepper Steps.Png. [imatge] Disponible a: <https://it.wikipedia.org/wiki/Filtro_mediano#/media/File:Salt_and_pepper_steps.png> [Accedit 28 Setembre 2020]. (CC BY-SA 3.0)

3. Desconegut, 2009. 512X512-Gaussian-Noise.Jpg. [imatge] Disponible a: <https://en.wikipedia.org/wiki/Gaussian_noise#/media/File:512x512-Gaussian-Noise.jpg> [Accedit 28 Setembre 2020]. (CC BY-SA 3.0)
4. Baccaglini, F., 2020. 4 Chromatic.Png. [imatge] Disponible a: <https://www.flickr.com/photos/185533101@N05/49352561877> [Accedit 28 Setembre 2020]. (Domini públic)
5. Wallace, R., 1904. Silver 'Grain' In Photography. [imatge] Disponible a: <https://commons.wikimedia.org/wiki/File:Film_Grain.jpg> [Accedit 9 Octubre 2020].
6. ComputerHoy, 2013. 4877-Ruido-Canon-Eos-5D-Mark-Iii. [imatge] Disponible a: <https://computerhoy.com/paso-a-paso/fotografia/que-es-como-calcular-ruido-foto-photoshop-1982> Accedit 25 Desembre 2020].
7. Drini. s/d. Triángulo de Pascal (versión sin rótulo). [imatge] Disponible a: < https://es.wikipedia.org/wiki/Tri%C3%A1ngulo_de_Pascal#/media/Archivo:Tri%C3%A1ngulo_de_Pascal.svg > [Accedit 21 Octubre 2020].

Fonts:

- 2000. Filtering And Enhancing Images. [ebook] Computer vision. Disponible a: <https://courses.cs.washington.edu/courses/cse576/book/ch5.pdf> [Accedit 21 octubre 2020].
- Abd Al-salam Selami, A. and Freidoon Fadhil, A., 2016. A Study Of The Effects Of Gaussian Noise On Image Features. [ebook] Kirkuk, Irak: Kirkuk University. Disponible a: <https://www.researchgate.net/publication/324476545_A_Study_of_the_Effects_of_Gaussian_Noise_on_Image_Features> [Accedit 10 Octubre 2020].
- Antoni, B., Lou, Y., Morel, J. and Tang, Z., 2010. Multi Image Noise Estimation And Denoising. [ebook] Disponible a: <https://www.researchgate.net/publication/45785986_Multi_image_noise_estimation_and_denoising> [Accedit 10 Octubre 2020].
- Arumugam, P., 2010. Image Denoising. [online] Codeding. Disponible a: <http://www.codeding.com/?article=10> [Accedit 21 octubre 2020].
- BROWN. 2020. Lab The Third. In Which It Is Shown How Noise May Be Overcome With Ye Bilateral Filter. [online] Disponible a: <https://cs.brown.edu/courses/cs129/labs/lab_bilateral/index.html> [Accedit 21 octubre 2020].
- Buades, A., Coll, B. and Morel, J., 2005. A Review Of Image Denoising Algorithms, With A New One. [ebook] Society for Industrial and Applied Mathematics. Disponible a: <https://www.researchgate.net/publication/41107019_A_Review_of_Image_Denoising_Algorithms_with_a_New_One> [Accedit 10 Octubre 2020].
- Butler, R., 2015. What's That Noise? Part One: Shedding Some Light On The Sources Of Noise: Digital Photography Review. [online] DPREVIEW. Disponible a: <https://www.dpreview.com/articles/8189925268/what-s-

that-noise-shedding-some-light-on-the-sources-of-noise> [Accedit 28 Setembre 2020].
- Cox, S., 2019. What Is Noise In Photography?. [online] Photographylife. Disponible a: <https://photographylife.com/what-is-noise-in-photography> [Accedit 28 Setembre 2020].
- Deng, H., 2009. MATHEMATICAL APPROACHES TO DIGITAL COLORIMAGE DENOISING. [ebook] Atlanta (EEUU): Georgia Institute of Technology. Disponible a: < https://smartech.gatech.edu/bitstream/handle/1853/31708/deng_hao_200912_phd.pdf > [Accedit 10 Octubre 2020].
- dev theomader. 2014. The Devil In The Details. [online] Disponible a: <http://dev.theomader.com/gaussian-kernel-calculator/> [Accedit 21 octubre 2020].
- Docs.Gimp. n.d. Desenfoque De Mediana. [online] Disponible a: <https://docs.gimp.org/es/gimp-filter-median-blur.html> [Accedit 21 octubre 2020].
- Ehmann, J., Chu, L., Tsa, S. and Liang, C., 2018. REAL-TIME VIDEO DENOISING ON MOBILE PHONES. [ebook] Google Inc. Disponible a: <http://chiakailiang.org/papers/ehmann_2018_rtv.pdf> Accedit 9 Octubre 2020].
- G. Schunck, B., Kasturi, R. and Jain, R., 1995. MACHINE VISION. [ebook] McGraw-Hill. Disponible a: <https://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter4.pdf> [Accedit 21 octubre 2020].
- Gombau, C., 2013. Qué Es Y Cómo Calcular El Ruido De Una Foto Con Photoshop. [online] Computerhoy. Disponible a: <https://computerhoy.com/paso-a-paso/fotografia/que-es-como-calcular-ruido-foto-photoshop-1982> [Accedit 9 Octubre 2020].
- In: DIEC2. n.d. soroll. [online] Barcelona: Institut d'Estudis Catalans. Disponible a: <https://dlc.iec.cat/Results?EntradaText=soroll> [Accedit 28 Setembre 2020].
- Irum, I., Alyas Shahid, M. and Raza, M., 2015. A Review Of Image Denoising Methods. [ebook] Islamabad: COMSATS University. Disponible a: <https://www.researchgate.net/publication/292145562_A_Review_of_Image_Denoising_Methods> [Accedit 9 Octubre 2020].
- Jain, M., Sharma, S. and Mohan Sairam, R., 2013. Effect Of Blur And Noise On Image Denoisingbased On PDE. [ebook] Jabalpur, India: Shri Ram Institute Of Technology. Disponible a: <https://www.semanticscholar.org/paper/Effect-of-Blur-and-Noise-on-Image-Denoising-based-Jain-Sharma/4eddffbb8fa8b3b928ea1a595a0a177681db6033> [Accedit 10 Octubre 2020].
- Libent Ishabailu, F., Pei, Z. and Mohamed Hambal, A., 2015. Image Noise Reduction And Filtering Techniques. [ebook] Tianjin, China: Tianjin University of Technology and Education. Disponible a: <https://www.ijsr.net/archive/v6i3/25031706.pdf> [Accedit 21 octubre 2020].

- Martí Pérez, F. i Pujalte Piñán, S. (2018). Histogrames i transformacions puntuals. [PDF] Universitat Oberta de Catalunya. Disponible a: https://www.uoc.edu [Accedit 11 Oct. 2019].
- Martí Pérez, F. i Pujalte Piñán, S. (2018). Transformacions espacials lineals. [PDF] Universitat Oberta de Catalunya. Disponible a: https://www.uoc.edu [Accedit 11 Oct. 2019].
- Mass-images.pro. n.d. Mass-Images.Pro - English Version. [online] Disponible a: <http://mass-images.pro> [Accedit 6 Novembre 2020].
- n.d. Técnicas De Realce De Imágenes. [ebook] DEA- Departamento de Electrónica y Automática, Facultad de Ingeniería, UNSJ. Disponible a: <http://dea.unsj.edu.ar/imagenes/recursos/capitulo3.pdf> [Accedit 21 octubre 2020].
- Navarro, P., 2017. El Grano Fílmico. [online] Medium. Disponible a: <https://medium.com/@paconavarro/el-grano-f%C3%ADlmico-609889f064d6> [Accedit 9 Octubre 2020].
- Paolo Farinella. 2019. Median Blending: The Best Noise Reduction Technique In Photoshop. [online] Disponible a: <https://www.paolofarinella.com/blog/2019/7/6/median-blending-the-best-noise-reduction-technique-in-photoshop> [Accedit 21 octubre 2020].
- Ryan, 2020. Median Filter를 사용하여 이미지의 Salt And Pepper Noise 제거 예제 코드. [online] ry.cl. blog. Disponible a: <https://ryanclaire.blogspot.com/2020/06/median-filter-salt-and-pepper-noise.html> [Accedit 21 octubre 2020].
- Sánchez Amador, S., s.d. Hipótesis Nula: Qué Es Y Para Qué Se Usa En Las Ciencias. [online] Psicología y mente. Disponible a: <https://psicologiaymente.com/cultura/hipotesis-nula> [Accedit 19 Desembre 2020].
- Sekhon, M., 2019. Image Filters In Python. [online] Towards data science. Disponible a: <https://towardsdatascience.com/image-filters-in-python-26ee938e57d2> [Accedit 21 octubre 2020].
- Sekhon, M., 2019. Image_Filters_In_Python.Ipynb. [online] Github. Disponible a: <https://github.com/m4nv1r/medium_articles/blob/master/Image_Filters_in_Python.ipynb> [Accedit 21 octubre 2020].
- Swain, A., 2018. Noise Filtering In Digital Image Processing. [online] Medium. Disponible a: <https://medium.com/image-vision/noise-filtering-in-digital-image-processing-d12b5266847c> [Accedit 21 octubre 2020].
- Wikipedia. 2020. Film Grain. [online] Disponible a: <https://en.wikipedia.org/wiki/Film_grain> [Accedit 9 Octubre 2020].
- Yusoff Mashor, M., Zamri Hasan, M., Naufal Mansor, M., Azman Wan Muhamad, W., Kadri Junoh, A. and Charmouti, B., 2017. Extended Median Filter For Salt And Pepper Noise In Image. [ebook] Kampus Pauh Putra, 02600, Arau, Perlis, Malaysia: Research India Publication. Disponible a: <https://www.ripublication.com/ijaer17/ijaerv12n22_156.pdf> [Accedit 21 octubre 2020].

- Zafra, D., n.d. What Is Noise In Photography And How To Get Rid Of It In 2020. [online] Capture the atlas. Disponible a: <https://capturetheatlas.com/noise-in-photography/> [Accedit 28 Setembre 2020].

Annex: Mètodes

En aquest apartat s'inclou la teoria sobre diferents mètodes de reducció de soroll que, o bé interpretant la lògica o bé adaptant el codi a Java, podrien ser útils per aquest anàlisi i aptes per el desenvolupament degut a la seva simplicitat.

Mètode 1: filtre bilateral gaussià.

"*Bilateral filter uses Gaussian Filter but it has one more multiplicative component which is a function of pixel intensity difference. It ensures that only pixel intensity similar to that of the central pixel is included in computing the blurred intensity value. This filter preserves edges.*" (Anisha Swain, 2018).

"*Kernel is a matrix with predefined values at each index. The result of convolution is a transformed image of the original image.*

Let us assume we have defined a 3-by-3 kernel, κ =

a b c

d e f

g h i

The first step of convolution is to flip the columns and rows of the kernel matrix. Let us represent the flipped matrix of κ by κf =

i h g

f e d

c b a

Note that for symmetric kernel, flipping rows and columns returns the original kernel matrix.

The next step is to slide this flipped matrix through the input image by placing the center of κf over each image pixel. When we do this, we replace the pixel value by multiplying kernel elements with pixel values directly below them and then taking the sum. If at a given instance, the following was the image matrix

below the sliding κf, when its center was placed over the image coordinates (x, y), denoted by ɪm =

p1 p2 p3

p4 p5 p6

p7 p8 p9

Then in the transformed image, at the pixel position (x, y) the pixel value is =

*i * p1 + h * p2 + g * p3 + f * p4 + e * p5 + d * p6 + c * p7 + b * p8 + a * p9*

From the above equation, we can derive that each image pixel value is replaced by the weighted sum of neighboring pixels with weights defined by flipped kernel elements over them. If ɪ denotes the input image matrix" (Mohit Sorde, 2017).

Més informació: veure filtre gaussià , algorisme 5.

Mètode 2: Crimmis spekle removal

"The Crimmins complementary culling algorithm is used to remove speckle noise and smooth the edges. It also reduces the intensity of salt and pepper noise. The algorithm compares the intensity of a pixel in a image with the intensities of its 8 neighbors. The algorithm considers 4 sets of neighbors (N-S, E-W, NW-SE, NE-SW.) Let a,b,c be three consecutive pixels (for example from E-S). Then the algorithm is:

1. *For each iteration:*
   a) Dark pixel adjustment: For each of the four directions
   1) Process whole image with: if a ≥ b+2 then b = b + 1
   2) Process whole image with: if a > b and b ≤ c then b = b + 1
   3) Process whole image with: if c > b and b ≤ a then b = b + 1
   4) Process whole image with: if c ≥ b+2 then b = b + 1
   b) Light pixel adjustment: For each of the four directions
   1) Process whole image with: if a ≤ b — 2 then b = b — 1
   2) Process whole image with: if a < b and b ≥ c then b = b — 1
   3) Process whole image with: if c < b and b ≥ a then b = b — 1
   4) Process whole image with: if c ≤ b — 2 then b = b — 1

Crimmins Speckle Removal

First a function that applies the Crimmins complementary culling algorithm will be defined.

In [14]:
def crimmins(data):

```
new_image = data.copy()
nrow = len(data)
ncol = len(data[0])

# Dark pixel adjustment

# First Step
# N-S
for i in range(1, nrow):
    for j in range(ncol):
        if data[i-1,j] >= (data[i,j] + 2):
            new_image[i,j] += 1
data = new_image
# E-W
for i in range(nrow):
    for j in range(ncol-1):
        if data[i,j+1] >= (data[i,j] + 2):
            new_image[i,j] += 1
data = new_image
# NW-SE
for i in range(1, nrow):
    for j in range(1, ncol):
        if data[i-1,j-1] >= (data[i,j] + 2):
            new_image[i,j] += 1
data = new_image
#NE-SW
for i in range(1, nrow):
    for j in range(ncol-1):
        if data[i-1,j+1] >= (data[i,j] + 2):
            new_image[i,j] += 1
data = new_image
# Second Step
# N-S
for i in range(1, nrow-1):
    for j in range(ncol):
        if (data[i-1,j] > data[i,j]) and (data[i,j] <= data[i+1,j]):
            new_image[i,j] += 1
data = new_image
# E-W
for i in range(nrow):
    for j in range(1, ncol-1):
        if (data[i,j+1] > data[i,j]) and (data[i,j] <= data[i,j-1]):
            new_image[i,j] += 1
data = new_image
# NW-SE
for i in range(1, nrow-1):
    for j in range(1, ncol-1):
        if (data[i-1,j-1] > data[i,j]) and (data[i,j] <= data[i+1,j+1]):
            new_image[i,j] += 1
data = new_image
# NE-SW
```

```
for i in range(1, nrow-1):
    for j in range(1, ncol-1):
        if (data[i-1,j+1] > data[i,j]) and (data[i,j] <= data[i+1,j-1]):
            new_image[i,j] += 1
data = new_image
#Third Step
# N-S
for i in range(1, nrow-1):
    for j in range(ncol):
        if (data[i+1,j] > data[i,j]) and (data[i,j] <= data[i-1,j]):
            new_image[i,j] += 1
data = new_image
# E-W
for i in range(nrow):
    for j in range(1, ncol-1):
        if (data[i,j-1] > data[i,j]) and (data[i,j] <= data[i,j+1]):
            new_image[i,j] += 1
data = new_image
# NW-SE
for i in range(1, nrow-1):
    for j in range(1, ncol-1):
        if (data[i+1,j+1] > data[i,j]) and (data[i,j] <= data[i-1,j-1]):
            new_image[i,j] += 1
data = new_image
# NE-SW
for i in range(1, nrow-1):
    for j in range(1, ncol-1):
        if (data[i+1,j-1] > data[i,j]) and (data[i,j] <= data[i-1,j+1]):
            new_image[i,j] += 1
data = new_image
# Fourth Step
# N-S
for i in range(nrow-1):
    for j in range(ncol):
        if (data[i+1,j] >= (data[i,j]+2)):
            new_image[i,j] += 1
data = new_image
# E-W
for i in range(nrow):
    for j in range(1,ncol):
        if (data[i,j-1] >= (data[i,j]+2)):
            new_image[i,j] += 1
data = new_image
# NW-SE
for i in range(nrow-1):
    for j in range(ncol-1):
        if (data[i+1,j+1] >= (data[i,j]+2)):
            new_image[i,j] += 1
data = new_image
# NE-SW
for i in range(nrow-1):
```

```
    for j in range(1,ncol):
        if (data[i+1,j-1] >= (data[i,j]+2)):
            new_image[i,j] += 1
data = new_image

# Light pixel adjustment

# First Step
# N-S
for i in range(1,nrow):
    for j in range(ncol):
        if (data[i-1,j] <= (data[i,j]-2)):
            new_image[i,j] -= 1
data = new_image
# E-W
for i in range(nrow):
    for j in range(ncol-1):
        if (data[i,j+1] <= (data[i,j]-2)):
            new_image[i,j] -= 1
data = new_image
# NW-SE
for i in range(1,nrow):
    for j in range(1,ncol):
        if (data[i-1,j-1] <= (data[i,j]-2)):
            new_image[i,j] -= 1
data = new_image
# NE-SW
for i in range(1,nrow):
    for j in range(ncol-1):
        if (data[i-1,j+1] <= (data[i,j]-2)):
            new_image[i,j] -= 1
data = new_image
# Second Step
# N-S
for i in range(1,nrow-1):
    for j in range(ncol):
        if (data[i-1,j] < data[i,j]) and (data[i,j] >= data[i+1,j]):
            new_image[i,j] -= 1
data = new_image
# E-W
for i in range(nrow):
    for j in range(1, ncol-1):
        if (data[i,j+1] < data[i,j]) and (data[i,j] >= data[i,j-1]):
            new_image[i,j] -= 1
data = new_image
# NW-SE
for i in range(1,nrow-1):
    for j in range(1,ncol-1):
        if (data[i-1,j-1] < data[i,j]) and (data[i,j] >= data[i+1,j+1]):
            new_image[i,j] -= 1
data = new_image
```

```
# NE-SW
for i in range(1,nrow-1):
    for j in range(1,ncol-1):
        if (data[i-1,j+1] < data[i,j]) and (data[i,j] >= data[i+1,j-1]):
            new_image[i,j] -= 1
data = new_image
# Third Step
# N-S
for i in range(1,nrow-1):
    for j in range(ncol):
        if (data[i+1,j] < data[i,j]) and (data[i,j] >= data[i-1,j]):
            new_image[i,j] -= 1
data = new_image
# E-W
for i in range(nrow):
    for j in range(1,ncol-1):
        if (data[i,j-1] < data[i,j]) and (data[i,j] >= data[i,j+1]):
            new_image[i,j] -= 1
data = new_image
# NW-SE
for i in range(1,nrow-1):
    for j in range(1,ncol-1):
        if (data[i+1,j+1] < data[i,j]) and (data[i,j] >= data[i-1,j-1]):
            new_image[i,j] -= 1
data = new_image
# NE-SW
for i in range(1,nrow-1):
    for j in range(1,ncol-1):
        if (data[i+1,j-1] < data[i,j]) and (data[i,j] >= data[i-1,j+1]):
            new_image[i,j] -= 1
data = new_image
# Fourth Step
# N-S
for i in range(nrow-1):
    for j in range(ncol):
        if (data[i+1,j] <= (data[i,j]-2)):
            new_image[i,j] -= 1
data = new_image
# E-W
for i in range(nrow):
    for j in range(1,ncol):
        if (data[i,j-1] <= (data[i,j]-2)):
            new_image[i,j] -= 1
data = new_image
# NW-SE
for i in range(nrow-1):
    for j in range(ncol-1):
        if (data[i+1,j+1] <= (data[i,j]-2)):
            new_image[i,j] -= 1
data = new_image
# NE-SW
```

```
for i in range(nrow-1):
    for j in range(1,ncol):
        if (data[i+1,j-1] <= (data[i,j]-2)):
            new_image[i,j] -= 1
data = new_image
return new_image.copy()"(Manvir Sekhon, 2019).
```

Veure exemple de la lògica en l'arxiu excel "Algorisme Crimmis spekle removal (logica).xlsx".

Mètode 3: desenfocament amb pes no uniforme

"2. Filtering with weighted moving average non-uniform weight

Previously we took the assumption that the true value of pixels are similar to true value of pixels nearby. But it is not always true. So for higher accuracy we assign the nearby pixels with greater weight then the pixels that are far away. This smooths the image and preserves the image information with less amount of data loss."(Anisha Swain, 2018).

"1. Filtering with weighted moving average uniform weight:

Instead of just thinking about averaging the local pixel, which is resulting in some loss of data, we consider a set of local pixel and assign them as uniform weights. Here we assume that noise is added to each pixel independently. According to this noise amount, we assign weights to different pixels.

2. Mean Filter:

Mean filter is a simple sliding window that replace the center value with the average of all pixel values in the window. The window or kernel is usually a square but it can be of any shape.

Fig.10 Calculating Mean by sliding window

2.1 Implementation of Mean Filter with OpenCV ans Python:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('poisson.jpg')

# kernel = np.ones((5,5),np.float32)/25
# dst = cv2.filter2D(img,-1,kernel)

#or
blur = cv2.blur(img,(5,5))

plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur),plt.title('Averaging')
plt.xticks([]), plt.yticks([])
cv2.imwrite('poissonfiltered.jpg',blur)
plt.show()
```

(Filtering Poisson Noise)"(Anisha Swain, 2018)

"import numpy as np
import cv2
from matplotlib import pyplot as plt
from PIL import Image, ImageFilter
%matplotlib inline
image = cv2.imread('AM04NES.JPG') # reads the image
image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV) # convert to HSV
figure_size = 9 # the dimension of the x and y axis of the kernal.
new_image = cv2.blur(image,(figure_size, figure_size))
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(cv2.cvtColor(image,
cv2.COLOR_HSV2RGB)),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(cv2.cvtColor(new_image,
cv2.COLOR_HSV2RGB)),plt.title('Mean filter')
plt.xticks([]), plt.yticks([])
plt.show()

We can check to see if any artifacts are created when a mean filter is applied to a gray scale image.

 # The image will first be converted to grayscale
image2 = cv2.cvtColor(image, cv2.COLOR_HSV2BGR)
image2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
figure_size = 9
new_image = cv2.blur(image2,(figure_size, figure_size))
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(image2, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(new_image, cmap='gray'),plt.title('Mean filter')
plt.xticks([]), plt.yticks([])
plt.show()"(Manvir Sekhon, 2019).

Mètode 4: filtre gaussià

"new_image = cv2.GaussianBlur(image, (figure_size, figure_size),0)
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(cv2.cvtColor(image, cv2.COLOR_HSV2RGB)),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_HSV2RGB)),plt.title('Gaussian Filter')
plt.xticks([]), plt.yticks([])
plt.show()

We can now check to see if the Gaussian filter produces artifacts on a grayscale image.

new_image_gauss = cv2.GaussianBlur(image2, (figure_size, figure_size),0)
plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(image2, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(new_image_gauss, cmap='gray'),plt.title('Gaussian Filter')
plt.xticks([]), plt.yticks([])
plt.show()" (Manvir Sekhon, 2019).

"Gaussian filters are a class of linear smoothing filters with the weights chosen according to the shape of a Gaussian function. The Gaussian smoothing fi.lter is a very good filter for removing noise drawn from a normal distribution. 1 The zero-mean Gaussian function in one dimension is

$$g(x) = e^{-\frac{x^2}{2\sigma^2}},$$

In two dimensions, Gaussian functions are rotationally symmetric. This means that the amount of smoothing performed by the filter will be the same in all directions. In general the edges in an image will not be oriented in sorne particular direction that is known in advance; con-sequently, there is no reason a priori to smooth more in one direction

2. The Gaussian function has a single lobe. This means that a Gaussian filter smooths by replacing each image pixel with a weighted average of the neighboring pixels such that the weight given to a neighbor decreases monotonically with distance from the central pixel. This property is important since an edge is a local feature in an image, and a smoothing operation that gives more significance to pixels farther away will distort the features.

3. The Fourier transform of a Gaussian has a single lobe in the frequency spectrum. This property is a straightforward corollary of the fact that the Fourier transform of a Gaussian is itself a Gaussian, as will be shown below. Images are often corrupted by undesirable high-frequency sig-nals (noise and fine texture). The desirable image features, such as edges, will have components at both low and high frequencies. The single lobe in the Fourier transform of a Gaussian means that the smoothed image will not be corrupted by contributions from unwanted high-frequency signals, while most of the desirable signals will be re-tained.

4. The width, and hence the degree of smoothing, of a Gaussian filter is pa.ra.meterized by O', and the relationship between O' and the degree of smoothing is very simple. A larger O' implies a wider Gaussian filter and greater smoothing. Engineers can adjust the degree of smoothing to achieve a compromise between excessive blur of the desired image features (too much smoothing) and excessive undesired variation in the smoothed image dueto noise and fine texture (too little smoothing).

5. Large Gaussian filters can be implemented very efficiently because Gaussian functions are separable. Two-dimensional Gaussian convolu-tion can be performed by convolving the image with a one-dimensional Gaussian and then convolving the result with the same one-dimensional filter oriented orthogonal to the Gaussian used in the first stage. Thus, the amount of cornputation required for a 2-D Gaussian fil ter grows lin-early in the width of tbe filter rnask instead of growing quadratically.

use row n of Pascal's triangle as a one-dimensional, n-point approximation to a Gaussian filter. For example, the five-point approxima-tion is:

1	4	6	4	1

corresponding to the fifth row of Pascal's triangle. This mask is used to smooth an image in the horizontal direction.

Another approach in designing Gaussian filters is to compute the mask weights directly from the discrete Gaussian distribution [146]:

$$g[i,j] = c\,e^{-\frac{(i^2+j^2)}{2\sigma^2}}$$

a two-dimensional Gaussian filter can be implemented as the successive convolutions of two one-dimensional Gaussians, one in the horizontal direc-tion and the other in the vertical direction.

$[i,j]$	-3	-2	-1	0	1	2	3
-3	1	4	7	10	7	4	1
-2	4	12	26	33	26	12	4
-1	7	26	55	71	55	26	7
0	10	33	71	91	71	33	10
1	7	26	55	71	55	26	7
2	4	12	26	33	26	12	4
3	1	4	7	10	7	4	1

This is the resulting convolution mask for the Gaussian fil ter. However, the weights of the mask do not sum to l. Therefore, when performing the convolution, the output pixel values must be normalized by the sum of the mask weights to ensure that regions of uniform intensity are not affected.

$$h[i,j] = \frac{1}{1115}(f[i,j] \star g[i,j])$$ " (Jain *et al.*, 1995).

Figura 7

Triángulo de Pascal

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1

Drini S/D. *Triángulo de Pascal (versión sin rótulo)* [imatge]. Wikipedia, Disponible a: https://es.wikipedia.org/wiki/Tri%C3%A1ngulo_de_Pascal#/media/Archivo:Tri%C3%A1ngulo_de_Pascal.svg

"# Gaussian convolution applied at each pixel in an image

 # Parameters:

 # im: Image to filter, provided as a numpy array

```
#   sigma:  The variance of the Gaussian (e.g., 1)
#
import numpy as np

def gaussian(im, sigma):
    height, width, _ = im.shape
    img_filtered = np.zeros([height, width, 3])

    # Define filter size.
    # A Gaussian has infinite support, but most of it's mass lies within
    # three standard deviations of the mean. The standard deviation is
    # the square of the variance, sigma.
    n = np.int(np.sqrt(sigma) * 3)

    # Iterate over pixel locations p
    for p_y in range(height):
        for p_x in range(width):
            gp = 0
            w = 0

            # Iterate over kernel locations to define pixel q
            for i in range(-n, n):
                for j in range(-n, n):
                    # Make sure no index goes out of bounds of the image
                    q_y = np.max([0, np.min([height - 1, p_y + i])])
                    q_x = np.max([0, np.min([width - 1, p_x + j])])
                    # Computer Gaussian filter weight at this filter pixel
                    g = np.exp( -((q_x - p_x)**2 + (q_y - p_y)**2) / (2 * sigma**2) )

                    # Accumulate filtered output
                    gp += g * im[p_y, p_x, :]
```

 # Accumulate filter weight for later normalization, to maintain image brightness

w += g

img_filtered[p_y, p_x, :] = gp / (w + np.finfo(np.float32).eps)

return img_filtered" (Desconegut, s/d).

Mètode 5: filtre conservatiu

"*El suavizado conservativo asegura que la intensidad del pixel está acotada dentro del rango de valores definidos por sus vecinos. Esto está acompañado por un procedimiento el cual primero encuentra los valores máximos y mínimos de todos los pixeles dentro de una ventana alrededor del pixel en cuestión. Si la intensidad del pixel central yace dentro del rango de intensidades definido por sus vecinos, el valor del pixel no se altera. Si por el contrario, la intensidad del pixel central es más grande que el máximo, se coloca el valor del máximo. Lo mismo para el mínimo. Ver figura 3.47*" (DEA- Departamento de Electrónica y Automática, Facultad de Ingeniería, UNSJ, s/d).

"Conservative Filter

The conservative filter is used to remove salt and pepper noise. Determines the minimum intensity and maximum intensity within a neighborhood of a pixel. If the intensity of the center pixel is greater than the maximum value it is replaced by the maximum value. If it is less than the minimum value than it is replaced by the minimum value. The conservative filter preserves edges but does not remove speckle noise.

The following code can be used to define a conservative filter:

```
# first a conservative filter for grayscale images will be defined.def
conservative_smoothing_gray(data, filter_size):temp = []

    indexer = filter_size // 2
    new_image = data.copy()
    nrow, ncol = data.shape
    for i in range(nrow):
        for j in range(ncol):
            for k in range(i-indexer, i+indexer+1):
                for m in range(j-indexer, j+indexer+1):
                    if (k > -1) and (k < nrow):
                        if (m > -1) and (m < ncol):
                            temp.append(data[k,m])
```

```
        temp.remove(data[i,j])
        max_value = max(temp)
        min_value = min(temp)
        if data[i,j] > max_value:
           new_image[i,j] = max_value
        elif data[i,j] < min_value:
           new_image[i,j] = min_value
        temp =[]
    return new_image.copy()
```

Now the conservative filter can be applied to a gray scale image:

new_image =
conservative_smoothing_gray(image2,5)plt.figure(figsize=(11,6))plt.subplot(121),
plt.imshow(image2, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(new_image, cmap='gray'),plt.title('Conservative Smoothing')plt.xticks([]), plt.yticks([])
plt.show()". (Manvir Sekhon, 2019)

Mètode 6**:** eliminació de sal i pebre

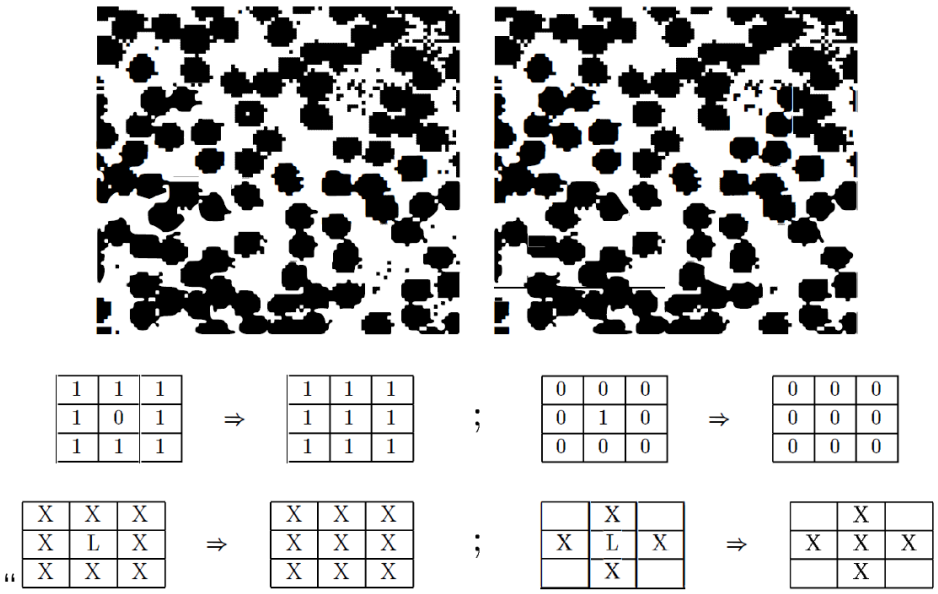


Figure 5.5: Binary image of red blood cells (top left) with salt and pepper noise removed

(top right). Middle row: templates showing how binary pixel neighborhoods can be cleaned.

Bottom row: templates defining isolated pixel removal for a general labeled input image;

(bottom left) 8-neighborhood decision and (bottom right) 4-neighborhood decision.

5.3.1 Removal of Salt-and-Pepper Noise

The introductory chapter briefly discussed how single anomalous pixels can be removed

from otherwise homogeneous regions and methods were extended in Chapter 3. The presence

of single dark pixels in bright regions, or single bright pixels in dark regions, is called

salt and pepper noise. The analogy to real life is obvious. Often, salt and pepper noise is

the natural result of creating a binary image via thresholding. Salt corresponds to pixels

in a dark region that somehow passed the threshold for bright, and pepper corresponds to

pixels in a bright region that were below threshold. Salt and pepper might be classification

errors resulting from variation in the surface material or illumination, or perhaps noise in

the analog/digital conversion process in the frame grabber. In sorne cases, these isolated

pixels are not classification errors at all, but are tiny details contrasting with the larger

neighborhood, such as a button on a shirt or a clearing in a forest, etc. and it may be that

the description needed for the problem at hand prefers to ignore such detail.

Figure 5.5 shows the removal of salt and pepper noise from a binary image of red blood

cells. The operations on the input image are expressed in terms of masks given at the

bottom of the figure. If the input image neighborhood matches the mask at the left, then

it is changed into the neighborhood given by the mask at the right. Only two such masks are needed for this approach. In case the input image is a labeled image created by use of a set of thresholds or sorne other classification procedure, a more general mask can be used. As shown in the bottom row of Figure 5.5, any pixel label 'L' that is isolated in an otherwise homogenous 8-neighborhood is coerced to the majority label 'X'. 'L' may be any of the k labels

used in the image. The figure also shows that either the eight or four neighborhood can be used for making the decision; in the case of the 4-neighborhood, the four comer pixels are not used in the decision. As shown in Chapter 3, use of different neighborhoods can result in different output images, as would be the case with the red blood cell image. For example, sorne of the black pixels at the bottom left of Figure 5.5 that are only 8-connected to a blood cell region could be cleaned using the 4-neighborhood mask." (Desconegut, 2000).

Mètode 7: barreja mediana

"If you opted for the last option (You take several shots of the same scene and then you apply the median filter in post) , here's an overview of how the median filter will come to rescue your image and save the day! But for this filter to perform at its best, you have to satisfy the following three conditions:

1. *The camera is on a tripod, so that frame does not change over time*
2. *The camera is in manual mode so that exposure does not change over time*
3. *You take at least 5 shots of the same scene, at the same exposure*

With all the 5 images stacked and aligned one on top of another, and for each given pixel location, there will be 5 different pixel values, because noise introduces pixel's value fluctuation.

In our example, we will have 5 values for the Red channel, 5 for the Green and 5 for the Blue, as follow:

- *Red (139, 145, 150, 151, 191)*
- *Green(166, 176, 200, 207, 215)*
- *Blue (49, 65, 87, 91, 100)*

By applying the median filter to the 5 values for each one of the RGB channels, we will end up with one single layer, with the following median pixel values:

- *RGB (150, 200, 87)*

Now, since neither the scene nor the exposure changed in all 5 images, the only thing affecting the value of a pixel id noise, and that is the key: by getting rid of that value fluctuation, we do get rid of noise itself.

How many images should you stack together to get the best result? I did run a few tests with stacks of 3, 5, 8 and 12 images and, as you can see in the graph below, I have noticed that the amount of noise decreases with the number of images stacked together, following a non linear curve: you will notice great improvements from a single shot to 3 stacked images but, as you add more

pictures to the stack, the improvements tend to be less noticeable." (Farinella, 2019).

Mètode 8: filtre de mediana ampliat per a soroll de sal i pebre

"Step 1:

In the first step we calculate the median "M" of the treated (central) pixel "TP" with its neighbors in the noisy image. Where this treated pixel may be noise or not. Figure 3 is presented as an example of this operation.

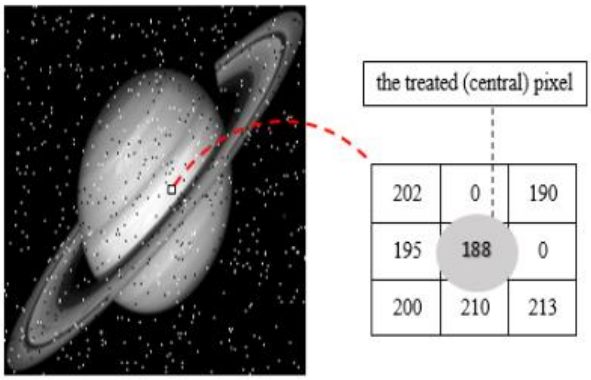


Figure 3: The Treated (Central) Pixel with Its Neighborhood.

In this example the treated pixel takes the value 188 (TP=188) as illustrated in Figure 3. Where the neighbors values are: 202, 0, 190, 195, 0, 200, 210, 213. In this case the median value is 195 (M = 195).

Step 2:

In the second step, the calculated median is taking as a reference and will be compared with the treated pixel to decide if this pixel should be changed by this median (which is 195 in this example) or keep the initial value 188. This comparison is made according to the following formula :

$abs(TP-M) \leq \delta$

Whre δ is a color-related parameter (positive integer). We choose δ when the addition of δ to the pixel's value, does not change the appearance of the color in this point.

If the formula (3) is realized, that means that the difference between the colors in the two points (treated pixel and median) is indistinguishable. In this case the treated pixel is considered as uncorrupted pixel and preserving its former value. Otherwise, this pixel is considered as a noise and should be replaced by the median. Where this operation is applied for all pixels in the noisy image. In our example if we replace TP and M by their values in (3)and if δ take the value 20 then the formula (3) is realized according to (4). Therefore, in this case the treated pixel keep the initial value 188.

$abs(TP-M)=abs(188-195)=7$

Usually in the case of others image filtering methods which adopt the noise detection phase in its processing, the filter is first look for the corrupted pixel (detection) and then look for witch value (alternate pixel) should replace this pixel, in order to treat only the corrupted pixels and keep others with same values (objective of noise detection phase). However, in our approach we look to achieve that objective with other way, we look first for the alternate pixel (median in this case) then decide to replace the treated pixel or keep it with the original value. This manner of detection is chosen to avoid the complex calculations occur in this phase and ensure that all corrupted pixels (noise) were captured.

Through this very simple proposed denoising technique we ensure that the uncorrupted pixels in the noisy image keep the same value after the denoising operation, and others corrupted pixels are changed by the most appropriate value (most closer to the original value). In order to get the most similar version to the original image." (Charmouti *et al.,* 2017).

Mètode 9: filtre de mediana

Aquest mètode és molt senzill d'aplicar i el seu funcionament sobre els píxels consisteix en atribuir a cada un, el valor resultant de la mediana dels seus veïns, eliminant així qualsevol rastre de píxels amb valors poc comuns, com poden ser els valors extrems del soroll impulsional.

"3. Median Filter:

Mean filter is a simple sliding window that replace the center value with the Median of all pixel values in the window. The window or kernel is usually a square but it can be of any shape.

Fig.11 Calculating Median by sliding window

3.1 Implementation of Median Filter with OpenCV and Python:

(Filtering Salt and Pepper noise)." (Swain, 2018).

"new_image = cv2.medianBlur(image, figure_size)plt.figure(figsize=(11,6))plt.subplot(121), plt.imshow(cv2.cvtColor(image, cv2.COLOR_HSV2RGB)),plt.title('Original')plt.xticks([]), plt.yticks([])plt.subplot(122), plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_HSV2RGB)),plt.title('Median Filter')plt.xticks([]), plt.yticks([])plt.show()." (Sekhon, 2019).

"*When a pixel is actually chosen from one of the white squares, but near the edge, it is likely that the majority of the values in its neighborhood are white pixels with noise: if this is true, then neighboring pixels from a black square will not be used to determine the output value.*

...

the median filter tends to preserve edge structure while at the same time smoothing uniform regions. Median filtering can also remove salt-and-pepper noise and most other small artifacts that effectively replace a few ideal image values with noise values of any kind." (Paul G. Allen School of Computer Science & Engineering, s/d).

"4.2.1 Advantage a)It is easy to implement. b)Used for de-noising different types of noises.

4.2.2 Disadvantage a)Median Filter tends to remove image details while reducing noise such as thin lines and corners. b)Median filtering performance is not satisfactory in case of signal dependent noise. To remove these difficulties different variations of median filters have been developed for the better results.

...

The main advantages of this medium filter are the denoising capability of the destroyed color component differences. Hence the method can be suitable for other filters available at present. But this technique increases the computational complexity. Our future research will be focused on the construction of other

Median filtering methods for color images to suppress other types of noises." (Mohamed H., 2015).

"The median-filter is also a sliding-window spatial filter, but the center value in the window is replaced by median (middle) of all the pixel values in the window. It is a more robust method than mean filter, because it is particularly effective in preserving the sharp edges in the image. Many variations to median filter are proposed such as weighted median filter, relaxed median filter, etc.

The following C# function illustrates the concept of a basic iterative 3x3 median filter.

view sourceprint?

```
01 public static PGM ApplyMedianFilter(PGM inputImage, int iterationCount)
02 {
03     PGM outputImage = new PGM(inputImage.Size.Width, inputImage.Size.Height);
04
05     for (int iteration = 0; iteration < iterationCount; iteration++)
06     {
07         for (int x = 0; x < inputImage.Size.Width - 1; x++)
08         {
09             for (int y = 0; y < inputImage.Size.Height - 1; y++)
10             {
11                 //get all neighbors of current-pixel
12                 List<int> validNeighbors = inputImage.GetAllNeighbors(x, y);
13
14                 //sort and find middle intensity value from all neighbors
15                 validNeighbors.Sort();
16                 int medianIndex = validNeighbors.Count / 2;
17                 short medianPixel = (short)validNeighbors[medianIndex];
18
```

```
            //set median-pixel as intensity of current-pixel
            outputImage.Pixels[x, y] = medianPixel;
        }
    }

    //set input to next iteration
    inputImage = outputImage.Clone();
}

return (outputImage);
}
```

" (Arumugam, 2010).

"The main problem with local averaging operations is that they tend to blur sharp discontinuities in intensity values in an image. An alternative approach is to replace each pixel value with the median of the gray values in the local neighborhood. Filters using this technique are called median filters.

Median filters are very effective in removing salt and pepper and impulse noise while retaining image details because they do not depend on values which are significantly different from typical values in the neighborhood. Median filters work in successive image windows in a fashion similar to linear filters. However, the process is no longer a weighted sum." (University of South Florida, s/d).

"Percentil

> *De forma predeterminada, el filtro de desenfoque de mediana encuentra el valor medio en el entorno de cada píxel. A pesar de su nombre, el filtro puede encontrar *cualquier* percentil arbitrario, no solo la mediana (es decir, el percentil 50). El parámetro «Percentil» controla el percentil utilizado para los componentes de color. Los valores más bajos cambian la imagen hacia tonos más oscuros y los valores más altos hacia tonos más brillantes*." (Gimp Documentation, s/d).

"typedef int element_type;

*int compare(const void *f1, const void *f2)*

```
{
 return (*(element_type*)f1 > *(element_type*)f2) ? 1 : -1;
}

element_type get_qsort_median(element_type*data, int n)
{
 if (!data || n <= 0) return 0;

 qsort(data, n, sizeof(element_type), compare);
 return data[n / 2];
}


int get_pixels(unsigned char *in, int *out, RECT *rect, int stride, int bytesperpixel)
{
 int x = 0;
 int y = 0;
 int n = 0;
 if (!in || !out || !rect) return -1;

 y = rect->top;
 while (y < rect->bottom)
 {
 x = rect->left;
 while ( x < rect->right)
 {
 int pos = (x*bytesperpixel) + (y*stride);
 out[n] = ((int)in[pos] & 0xff) << 16 | ((int)in[pos+1] & 0xff) << 8 | ((int)in[pos+2] &
0xff);
 n++;
 x++;
 }

 y++;
 }

 return 0;
}

int median_filter_2d(unsigned char *in, unsigned char *out, int W, int H, int
BytesPerPixel)
{
#define window_size 3
 int temp[window_size*window_size] = { 0, };
 int x = 0;
 int y = 0;
 int stride = W*BytesPerPixel;

 if (!in || !out || W < window_size || H < window_size) return -1;

 y = 0;
```

```
while (y < H)
{
x = 0;
while (x < W)
{
RECT rect;
int pos = x*BytesPerPixel + y*stride;
int val = 0;

rect.left = x - window_size / 2;
rect.top = y - window_size / 2;
if (rect.left < 0) rect.left = 0;
if (rect.top < 0) rect.top = 0;

rect.right = rect.left + window_size;
rect.bottom = rect.top + window_size;

if (rect.right >= W) rect.right = W - 1;
if (rect.bottom >= H) rect.bottom = H - 1;

get_pixels(in, temp, &rect, stride, BytesPerPixel);
val = get_qsort_median(temp, (rect.right - rect.left)*(rect.bottom - rect.top));
out[pos] = (val & 0xff0000) >> 16;
out[pos+1] = (val & 0xff00) >> 8;
out[pos+2] = (val & 0xff);

x++;
}
y++;
}

return 0;
}
```
" (Ryan, 2020).

Mètode 10: filtre mitjà

"Mean filter is a simple and intuitive way to reduce the image noise. It is a sliding-window spatial filter that replaces the center value in the window with the average (mean) of all the pixel values in the window. The idea is to eliminate the pixel values which are unrepresentative of their surrounding pixels. The window size is usually 3x3. Larger window sizes or repeated (iterative) mean filters introduce blurring and other unwanted effects in the image data.

The following C# function illustrates the concept of a basic iterative 3x3 mean filter. The GetAllNeighbors() function returns all valid pixels in the 3x3 window of current pixel. Invalid pixels are those that does not exist; for example corner pixels have only four neighbors including itself.

```
public static PGM ApplyMeanFilter(PGM inputImage, int iterationCount)
{
    PGM outputImage = new PGM(inputImage.Size.Width, inputImage.Size.Height);

    for (int iteration = 0; iteration < iterationCount; iteration++)
    {
        for (int x = 0; x < inputImage.Size.Width - 1; x++)
        {
            for (int y = 0; y < inputImage.Size.Height - 1; y++)
            {
                //find average intensity of neighbors of current-pixel
                List<int> validNeighbors = inputImage.GetAllNeighbors(x, y);
                double averageIntensity = validNeighbors.Average();

                //set average as new intensity of current-pixel
                outputImage.Pixels[x, y] = (short)averageIntensity;
            }
        }

        //set input to next iteration
        inputImage = outputImage.Clone();
    }

    return (outputImage);
}
```

" (Arumugam, 2010).

"One of the simplest linear filters is implemented by a local averaging opera-tion where the value of each pixel is replaced by the average of all the values in the local neighborhood:

$$h[i,j] = \frac{1}{M} \sum_{(k,l)\in N} f[k,l]$$

where Mis the total number of pixels in the neighborhood N. For example, taking a 3 x 3 neighborhood about [i, j] yields:

$$h[i,j] = \frac{1}{9} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} f[k,l].$$

" (University of South Florida, s/d).