# Introducere

## Contextul proiectului

Proiectul se dorește a oferi o modalitate eficienta de extragere si combinare a informației relevante din mai multe surse de imagini medicale prin fuziunea lor. Prin acest procedeu se poate vizualiza si prognoza o eventuala boala, tumoare sau orice alta disfuncțiune a organismului.

Rolul principal al acestui procedeu este de a scuti pacientul de mai multe metode de imagistica medicala, invazive pentru corp, cum ar fi Rezonanta Magnetica (RMN), Computer Tomograf (CT). Aceste tehnologii, deși deseori oferă rezultate mai precise sau sunt mai recomandate pentru anumite părți ale corpului, cu utilizare repetata pot cauza probleme medicale pentru cel pe care se aplica. Ideea generala este de a efectua o singura data una din metodele amintite mai sus, la un stadiu initial al starii de sanatate al pacientului, dupa care se pot face mai multe imagini non-invazive, cum ar fi ecografia. Combinand cele doua tipuri de imagine, se va rezulta o imagine de calitate mai buna decat oricare cele doua imagini sursa, si care va simboliza zona corpului intr-o stare cat mai recenta.

Aceasta metoda de fuziunea imaginilor medicale se practica deja in tarile vestice, precum si Franta, in capitala careia, Paris, se organizează anual conferința ICIFE (International Conference on Image Fusion Engineering), care are ca scop insumarea experientelor si rezultatelor cercetarii in domeniul fuziunii imaginilor.

Scopul proiectului este de a oferi o soluție la îmbunătățirea informației obținută din mai multe imagini medicale, care au fost făcute utilizând metode diferite, si au fost efectuate la stadii diferite al stării de sănătate al pacientului. Se vor folosi diferiți algoritmi pentru fuziunea propriu-zisa, si se vor compara rezultatele primite atât subiectiv, cu ajutorul a mai multor voluntari care oferă anumite note pentru aspectul imaginilor, cat si obiectiv, cu ajutorul unor metode de măsurarea calității imaginilor.

## Conturarea domeniului exact al proiectului

Imagistica medicala este o ramura a ingineriei biomedicale. Prin diferite tehnici de inregistrarea imaginilor medicale asupra organelor si tesuturilor, se pot diagnostica mai usor bolile in organismele vii. Se pot folosi atat in domeniul clinical cat si in cel stiintific. Exista mai multe instrumente pentru efectuarea acestor procedee care au propriile avantaje si dezavantaje. Se folosesc proprietati chimice sau fizice care pot induce crearea imaginii medicale in scopul de a obtine date importante. Datele pot fi codificate in mai multe moduri: imagini 2D, 3D, imagini de spectru sau liste de valori.

Acest sistem se dorește a ajuta tehnologia medicala sa ajungă la cel puțin nivelul dispozitivelor mobile de astăzi, accentuând faptul ca si domeniul medical necesita dezvoltare, avansare condusa de persoane pasionate. Obiectivul final este limitarea timpului pierdut, care de multe ori poate fi crucial in analizarea si prognosticarea imaginilor medicale.

Telul aplicatiei este de a crea, dintr-o imagine RMN sau CT si o imagine de tip ecografie, o a treia imagine, care contine informatii mai relevante decat oricare din primele doua. Pentru acesta se vor aporda urmatoarele clase de metode de fuziune si cinci implementări in total:

* Metode aritmetice
  + Valoarea minima
  + Valoarea maxima
  + Valoarea medie
* Metode piramidale
  + Piramida Laplaciana
* Metode bazate pe transformata Wavelet
  + Transformata Discreta Wavelet Haar

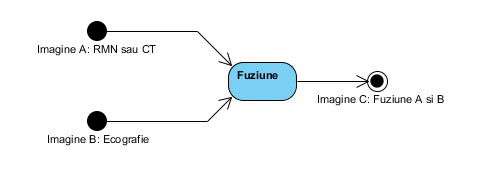


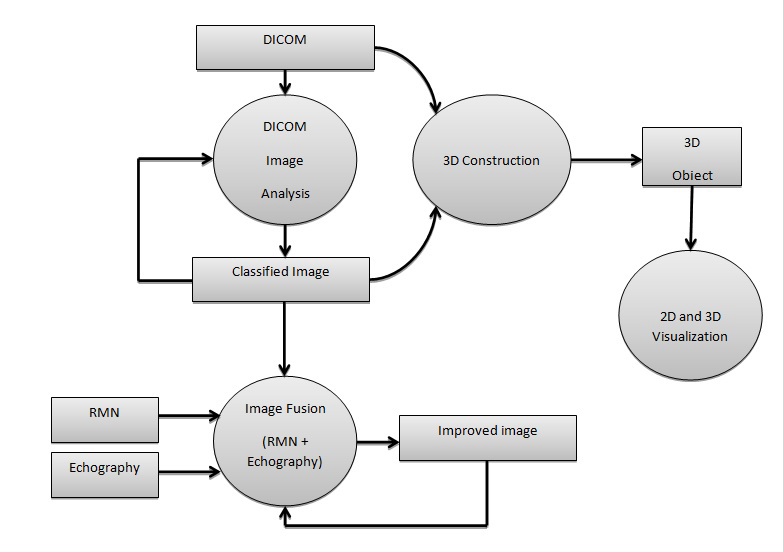
Figura 1.2: Fuziunea imaginilor

Pentru a compara aspectul vizual al rezultatelor, se vor ruga 10 voluntari sa acorde note de la 0 la 10 pentru fiecare imagine rezultat, fara sa stie in prealabil ce algoritm s-a folosit pentru calcularea acestuia. Imaginile rezultate se vor putea compara cu o imagine „perfecta”. Langa aceata masurare mai mult subiectiva, se vor aplica si doi metrici de masurarea calitatii imaginilor:

* Eroarea medie patratica (Mean squared error – MSE)
* Raportul intre semnalul de var si zgomot (Peak signal to noise ratio – PSNR)

# Obiectivele proiectului

Proiectul face parte dintr-un proiect mai complex, de analizarea, fuziunea si reconstrucția imaginilor medicale atât in 2D, cat si in 3D, din care abordează partea de fuziune a imaginilor (modulul de Image Fusion din imaginea de mai jos).



*Figură 2.1 Proiectul de analiza, fuziunea si reconstructia imaginilor medicale*

Scopul acestui proiect este de crea o aplicatie care serveste doua functionalitati: fuziunea imaginilor medicale si studiul diferiților algoritmi pentru a efectua acest lucru. Aceste doua parti nu sunt complet independente, compararea algoritmilor folosind modulul de fuziune, care contine si logica de procesarea a imaginilor.

Necesitatea acestei aplicatiei consta in faptul extragerea si combinarea informatiei din doua surse diferite, dar care contin date despre acelasi obiect/subiect (in cazul nostru partea de corp al aceluiasi pacient), poate ajuta la o dioagnoza mai precisa. Pe de alta parte, minimizand expunerea unui pacient la metode cu raze intruzive, dar cu rezolutie mai buna (cum ar fi metoda RMN sau CT), este benefic din punctul de vedere al sanatatii acestuia.

Conform ideii de mai sus, procesul de imagistica medicala s-ar transforma in aplicarea unei metode de rezolutie superioara, dar intruziva, la inceputul stagiului de consultanta, dupa care se fac mai multe imagini de calitate mai redusa, dar non-intruzive pentru corpul uman. Dupa fuziunea imaginii initiale cu imaginea facuta intr-un stadiu mai avansat, va rezulta o a treia imagine, cu informatii mai precise si mai actuale decat oricare din sursele precedente. Prin acest mod s-ar reduce efectul negativ cauzat de metodele intruzive, dar si timpul petrecut la centre medicale, metodele RMN si CT avand durata de aplicare foarte mare (spre exemplu, un RMN pentru o parte relativ mica a corpului, umarul, dureaza 50 de minute, si este si incomfortabil).

Bazandu-se pe modulul de fuziune propriu zisa de mai sus, se proiecteaza si un modul de comparare a diferitilor algoritmi, din punctul de vedere a calitatii imaginii de iesire, folosind mai multe metrici de masurare a calitatii. Aceasta masurare se doreste a fi atat obiectiva, creand un proces algoritmic care va avea ca iesiri valorile pentru calitatea imaginilor, cat si subiectiv, care va consta in analizarea imaginilor rezultat de catre mai multi voluntari, acordandu-le note.

Ca parametri de intrare, aplicatia de fuziune foloseste in principal fisiere DICOM, pentru ca imaginile medicale moderne sunt codificate conform acestui protocol. Optional, se pot incarca si imagini de format jpg, png sau gif, in scopul in care utilizatorul vrea sa testeze diferite modificari sau rezultate. Modulul de masurarea calitatii algoritmilor va folosi ca intrare o list imagini .jpg deja incluse in proiect.

## Cerinte functionale

1. Incarcarea a doua imagini de tip DICOM.

Aplicatia trebuie sa fie capabila sa incarce doua fisiere de tip DICOM, care sa contina date despre imaginile medicale. Acestea vor fi extrase pentru a putea realiza fuziunea.

1. Afisarea imaginilor incarcate

Aplicatia trebuie sa afiseze informatiile de imagine extrase din fisierele DICOM in ferestre noi. Pentru o stiva de imagini, trebuie sa adauge o optiune de a vizualiza toate imaginile din stiva.

1. Customizarea parametrilor algoritmului de fuziune.

Parametri algoritmilor de fuziune, care vor afecta rezultatul procesului, trebuie sa fie customizabile din interfata utilizatorului.

1. Customizarea metodei de post-procesare.

Utilizatorul trebuie sa aiba posibilitatea sa aleaga printre mai multe metode de postprocesare a rezultatului, printre care trebuie sa fie prezenta si optiunea de non-postprocesare (afisarea rezultatului „crud” de dupa fuziune).

1. Afisarea rezultatelor in ferestre.

Aplicatia trebuie sa afiseze rezultatele fuziunii in ferestre noi, pastrand ferestrele cu imaginile de intrare, precum si rezultatele precedente ale fuziunii.

1. Salvarea rezultatelor la alegerea utilizatorului.

Aplicatia trebuie sa ofere posibilitatea de salvare a rezultatelor procesului de fuziune, in format de imagine obisnuit (.jpg).

1. Afisarea unei pagini de ajutor, instructiuni.

Aplicatia trebuie sa afiseze o pagina de ajutor cu pasii principali si sfaturi asupra utilizarii sistemului.

1. Rularea metodei de masurarea calitatii a algoritmilor.

Aplicatia trebuie sa-i ofere posibilitatea utilizatorului sa ruleze procesul de masurarea calitatii a algoritmilor de fuziune.

1. Customizarea listei de algoritmi pentru acesta

Aplicatia trebuie sa ofere o posibilitate de a alege in ce format se doreste salvarea rezultatelor procesului de masurarea calitatii.

1. Salvarea rezultatelor in formate alese de utilizator.

Aplicatia trebuie sa salveze in fisiere separate, pe masina locala, rezultatele procesului de masurare a calitatii.

## Cerinte non-functionale

1. Viteza

Aplicatia trebuie sa efectueze pe un sistem de calcul performant, fuziunea imaginilor, in mai putin de 5 secunde. Aceasta viteza va depinde si de putearea de calcul a masinii pe care se ruleaza programul. Procesul de masurarea calitatii sa nu dureze mai mult de 10 secunde, incluzand si salvarea rezultatelor in fisiere.

1. Precizie

Aplicatia trebuie sa produca rezultate consistente, pentru aceleasi valori de intrare, atat pentru fuziune cat si pentru masurarea calitatii.

1. Functionare independanta de platforma

Aplicatia trebuie sa functioneze pe orice sistem care are Java Runtime Environment instalat. Nu include componente care tin de o singura platforma.

1. Portabilitate

Aplicatie trebuie sa fie portabila. Prin acest lucru se intelege faptul ca ajunge ca sa se copieze fisierul executabil al aplicatiei pe alt sistem, fara a fi nevoie de configurari specifice.

1. Functionare offline

Aplicatia trebuie sa functioneze la orice moment dat, fara a avea nevoie de o conexiune la internet.

## Partile esentiale

Partea cea mai complexa si importanta din punctul de vedere al functionarii corecte a aplicatiei este partea algoritmica, care implementeaza algoritmii. Acestia sunt folositi la fuziunea propriu-zisa a doua matrici de informatii codificate folosind numere intregi, care in cazul nostru vor fi reprezentate de matrici de pixeli. Aceasta componenta defineste logica de baza a functionarii aplicatiei, si pe ea se bazeaza in mare parte si celelalte module.

Componenta de masurarea calitatii este importanta din punct de vedere al compararii algoritmilor mentionati mai devreme, folosind moduri de verifacarea calitatii obiective. Ea va itera peste algoritmii implementati in modulul de mai sus, folosindu-le la fuziunea imaginilor cu mai multe optiuni, pentru a primi o gama larga de rezultate.

Componenta care leaga cele doua amintite mai sus, este interfata grafica. Ea ofera o legatura intre logica aplicatiei si utilizator, folosind un mod de vizualizare si interactiune simpla si intuitiva. La randul sau, si interfata expune diferitele optiuni pentru rularea procesului de fuziune sau metodei de masurarea calitatii.

### Datele de intrare si iesire

Aplicatia va permite ca date de intrare trimise de catre utilizator, fisiere de tip DICOM (.dcm sau alte formate specifice). E important de menționat faptul ca unele fisiere dicom compuse dintr-o stiva de imagini, nu au nici o extensie de fisier. Aplicatia accepta si imagini de acest tip, verificandu-le structura corecta pentru a fi procesate pe urma. La fel, din motive de testare, depanare, se pot incarca si imagini de format obisnuit (.jpg, .png, .gif).

Ca date de iesire, se primesc imagini fuzionate care se pot vizualiza in ferestre, si se pot salva in format .jpg. Rezultatele modulului de masurarea calitatii vor fi salvate ca text (.txt) sau ca fisier Excel (.xls).

# Studiu Bibliografic

## Domeniul in care se situeaza tema

Aplicatia avand scopul imbunatatirii metodelor de diagnoza medicala, trebuie studiata mai in detaliu acest domeniu.

### Imagistica medicala

In articolul [[1]](#endnote-1) scrie ca Imagistica medicala este un domeniu stiintific relative recent, al carui scop principal este extragerea informatiilor importante din organisme vii, folosind metode fizice sau chimice.

In capitolul 2, subcapitolul 2.1 din [[2]](#endnote-2), se mentioneaza urmatoarele metode de imagistica medicala: radioscopia, radiografia, tomografia, ultrasonografia, imagistica prin rezontanta magnetica si tomografia prin emisie de pozitroni. Dintre acestea, unele metode aduc si riscuri cu ele din punctul de vedere al sanatatii pacientului.

Radioscopia, desi este metoda cea mai ieftina, poate iradia bolnavul. O alternativa mai buna la aceasta este radiografia, care insa necesita numeroase filme pentru a urmari precis functionarea unor organe. Ultrasonografia desi e cea mai sigura, produce rezultate care sunt mai greu de interpretat, din cauza faptului ca reprezinta organele interne intr-o forma nenaturala. Tomografia este mai ieftina decat imagistica prin rezonanta magnetica, este superioara pentru imagini ale craniului uman, dar aduce la fel radiatii cu ea, deci nu poate fi aplicata la pacienti cum ar fi femeile gravide sau bolnavi cu diabet. Rezonanta magnetica este probabil cea mai scumpa metoda, este foarte precisa. Procesul in sine poate cauze sentiment de claustrofobie, este lung si sensibil la miscari. La fel, undele magnetice sunt invazive pentru corpul uman.

Luand in considerare avantajele si dezavantajele prezsentate mai sus, putem ajunge la concluzia ca nu exista o singura metoda care e perfecta din toate punctele de vedere. Din aceasta cauza, fuziunea rezultatelor de la diferite surse de imagistica medicala ar putea fi solutia cea mai avantajoasa si pentru bugetul pacientului, dar si din punctul de vedere al diagnosticului corect.

### Standardul DICOM

Din articolul [[3]](#endnote-3), putem afla ca Standardul DICOM (Digital Imaging and Communications in Medicine) este folosit pentru stocarea, comprimarea si transmiterea datelor medicale. Acest standard defineste modul de stocare fizic a informatiilor si un protocol de comunicare pe reatea, bazat pe TCP/IP. Fisierele DICOM contin, de obicei, date in fomat textuat despre pacient si mediul de efectuarea imaginii, si una sau mai multe imagini.

Datele sunt combinate in asa fel, incat numarul identificator al pacientului nu se poate separa de imaginea (imaginile) lui. Partea textuala a fisierului poate contine mai mult de 3300 de tag-uri specifice imaginii, pacientului sau instrumentului folosit, numite si atribute. Cateva exemple de atribute:

* Imagine
  + Datele continute de pixeli
  + Lista de frame-uri
  + Inceputul decuparii
  + Sfarsitul decuparii
  + Tipul frame-ului
  + Proprietati volumetrice
* Pacient
  + Numele pacientului
  + Numarul sau identificator
  + Varsta
  + Grupa sanguina
  + Alergii
  + Grupa etnica
  + Fumator
  + Tipul terapiei
* Instrumentul folosit
  + Firma producatoare
  + Numarul identificator al aparatului
  + Modalitatea efectuarii imaginii
  + Numele institutului
  + Adresa institutului

Cateva dintre tag-uri pot aparea de mai multe ori in fisier. Imaginea propriu zisa este reprezentata in tag-ul de datele pixelilor. Desi acest atribut poate fi prezent doar o data in fisier, putem avea mai multe imagini asociate cu un DICOM. Diferenta intre imagini se face prin precizarea numarului si asezarii frame-urilor.

Pentru a usura si a optimiza afisarea imaginilor pe dispozitive diferite, standardul defineste un tabel de cautare dupa valoarea pixelilor DICOM Grayscale Standard Display Function. Pentru a putea afisa corect datele continute de pixeli, dispozitivele trebuie sa contina aceasta functie sau sa fie calibrate pentru acest fel de afisare.

Pentru comunicare, standardul DICOM foloseste TCP sau UDP si portul 104. Standardul defineste si o serie de servicii pentru a usura comunicarea pe retea: stocare, cautare, listare si definirea unei operatii automate de imagistica.

Una din dezavantajele formatului DICOM, este considerat a fi posibilitatea de a definii prea multe atribute optionale si completarea unor campuri cu informatii gresite. Prin acest fel, se pierde din consistenta combinatiei text-imagine.

### Fuziunea imaginilor

In articolul [[4]](#endnote-4), putem gasi definitia acestei tehnologii teoretice. Fuziunea imaginilor este procesul prin care se combina datele relevante din doua sau mai multe imagini sursa intr-o singura imagine, in care rezultatul va contine informatii mai complete si mai precise decat oricare din imaginile de intrare. In acest mod, fuziunea imbunatateste calitatea rezultatelor pentru un anumit domeniu.

In medicina [[5]](#endnote-5), fuziunea imaginilor devine din ce in ce mai folosit pentru a imbunatati efectele si a reduce durata tratamentelor. Imaginile fuzionate pot fi create folosind ca intrare rezultatele aceleiasi tehnologie de imagistica, sau combinand modalitatile de achizitionare a datelor despre pacient. Imaginile rezultate, consistente in informatiile pe care le expun, sunt importante mai ales pentru detectia cancerului in corpul uman.

## Aplicatii similare

Aplicatii prin care se poate efectua fuziunea imaginilor medicale:

Mirada Medical XD3 [[6]](#endnote-6), care printre altele, ofera unelte pentru vizualizarea, inregistrarea manipularea si segmentarea diferitelor imagini.

Velocity Medical [[7]](#endnote-7), special conceput pentru oncologie, ofera integrarea usoara a imaginilor de la tratamente diferite, cu este radioterapia si brachoterapia.

Keosys Imagys-Cloud Services [[8]](#endnote-8), pune la dispozitie o aplicatie de vizualizarea si fuziunea imaginilor medicale 3D.

## Framework-uri folosite

In cele ce urmeaza, se prezinta lista framework-urilor dintre posibilitatile care s-au studiat, si care s-au dovedit cele mai potrivite pentru acest proiect.

### ImageJ

ImageJ este o aplicatie de procesare si analizare a imaginilor, scrisa in limbajul Java. Acesta [[9]](#endnote-9) aduce cu sine si indepedenta de platforma, programul putand fi rulat atat pe sisteme Linux, cat si pe Max OSX si Windows, 32 sau 64 de biti. Programul si codul sau sursa este open-source, fara a avea nevoie de o licenta. Este extensibil prin framework-ul robust si documentat, oferit de echipa dezvoltatoare, prin care se pot crea noi plugin-uri si programe de sine statatoare folosind aceasta tehnologie. In momentul de fata exista deja 500 de plugin-uri pentru ImageJ. Pe site-ul oficial, se sustine faptul ca ImageJ este cel mai rapid program de procesarea imaginilor, scris in Java. Programul poate deschide si salva imagini de formatul GIF, JPEG, BMP, PNG, PGM, FITS si poate deschide fisiere DICOM, care este folositor pentru scopul nostru.

Programul suporta modificarea si filtrarea imaginilor deja deschise, punand la dispozitie cateva module deja definite, dar permitand si integrarea functionalitatilor noi in aplicatie. Poate procesa imagini pe 8, 16 sau 32 de biti. Se pot aplica operatii de masurare a diferitelor suprafete, lungimi si unghiuri.

Printre altele, framework-ul ofera si functionalitate de citirea imaginilor si stivelor de imagini in format DICOM, prelucrarea lor, extragerea informatiilor din aceste fisiere, si separarea lor in date despre pacient si imaginea efectiva.

### JExcel API

Acesta este un modul Java integrabil [[10]](#endnote-10), care pune la dispozitie functii de citire, scriere si modificare a fisierelor Excel. Suporta formate de Excel 95, 97, 2000, XP si 2003. Poate citi si scrie si formule. Genereaza foi de calcul si suporta formatul de litera, numar sau data calendaristica. Pe langa astea, ofera o interfata logica si usor de inteles pentru dezvoltatori, includerea lui in proiecte fiind la fel usoara.

### Java Swing

Swing este un framework de proiectare a interfetelor grafice de utilizator pentru limbajul Java. Din articolul [[[11]](#endnote-11)] gasim modurile in care se poate folosi acesta. Se alege un cadru, „JFrame”, care se poate popula cu diferite elemente vizuale, cum ar fi campuri de date de intrare, butoane, liste, meniuri si poze sau alte animatii. Pe aceste elemente se pot scrie diferite actiuni, care pot interactiona cu logica din spatele aplicatiei. Fiecare element vizual poate intercepta actiunile utilizatorului in mai multe feluri, fie apasarea unei taste, click-ul pe element sau scroll-ul asupra paginii. Acestea se proceseaza de catre manipulantii de evenimente (EventHandler-uri), care sesizeaza actiunea ce a avut loc si fac operatiile ce sunt scrise pe actiunea respectiva.

### JUnit

Pentru a asigura calitatea si functionarea corecta a sistemului, se recomanda a folosi teste de unitate pentru verificarea raspunsului modulelor aplicatiei la diferite date de intrare si in diferite conditii. JUnit, [[12]](#endnote-12), este un framework de testare pentru limbajul de programare Java. Este estential in metodele de dezvoltare bazate pe teste (test drive development) [[13]](#endnote-13). Folosind aceasta unealta, se doreste a automatiza testarea aplicatiei cat mai in detaliu, in acest mod garantand comportamenul corect al acesteia.

### Maven

Maven [[14]](#endnote-14), este o unealta pentru administrarea, automatizarea si configurarea procesului de build a unei aplcatii Java. Pe langa astea, poate fi folosit si pentru a defini dependintele unei aplicatii, pe care le poate descarca de pe un server global, si le integreaza in proiect. Unealta se configureaza dintr-un fisier XML numit „pom.xml”, prin care se poate preciza ordinea proceselor de construire a codului aplicatiei, cum vor fi rulate testele, cum va arata rezultatul compilarii si in ce structura de foldere va fi pus.

In cazul nostru, modulele ImageJ, JExcel API si JUnit vor fi declarate si descarcate folosind Maven, iar testele vor fi automatizate tot din script-ul scris in „pom.xml”. Asta ofera o independenta a proiectului fata de editorul de cod pe care-l folosim, procesul de build avand comportament identic pe orice platforma si editor.

## Concluzie

Studiind aceste tehnologii stiintifice si tehnice, aplicatia software va beneficia aplicandu-le in cadrul ei. Probabil se va pune intrebarea, de ce s-a ales limbajul Java pentru construirea aplicatiei, si nu limbajul C sau C++, care s-au dovedit a fi superioare din punctul de vedere a vitezei de procesare in general, si mai ales in cazul procesarii imaginilor.

In primul rand, framework-ul ImageJ ne pune la dispozitie o serie de module deja implementate, care ne ajuta in citirea fisierelor DICOM, prelucrarea acestora si procesarea imaginilor extrase din ele. Scopul aplicatiei nu este de a avea o structura unica, implementata de la zero, ci de a folosi modulele deja existente, cuplandu-le in asa fel incat sa ofere o solutie cat mai eficienta la rezolvarea unor anumite probleme, in cazul nostru, fuziunea imaginilor medicale. Aplicatia in sine este scrisa avand in vedere atributul de a fi reutilizabil in viitor, metodele si clasele fiind extensibile si configurabile usor.

In al doilea rand, avand interfata si modulele de procesare din spate in Java, aplicatia poate fi rulata pe orice platforma. O solutie scrisa, de exemplu, in C++ cu domeniul de prezentare implementat in C#, ar fi avut constragerea sa fie rulat doar pe sisteme Windows. Proiectul se doreste a fi open-source, scopul principal fiind ca alti dezvoltatori sau cercetatori sa-l integreze cu usurinta pe platforma lor in care lucreaza cel mai eficient.

Un alt aspect, la fel relativ important, este experienta deja dobandita in cadrul facultatii, in limbajul Java. Prin acest mod, dezvoltarea efectiva aplicatiei a rezultat intr-un ritm mai productiv de scriere a codului, pe care s-au aplicat si practicile cele mai bune invatate la cursuri si la laboratoare. La fel, integrarea si folosirea framework-urilor alese a fost mai facila, tot din cauza experientei dobandite deja la numeroase proiecte in Java.

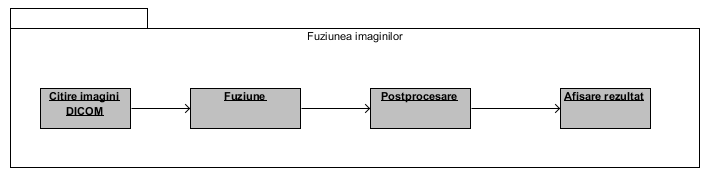
1. <http://en.wikipedia.org/wiki/Image_fusion>
2. <http://asp.eurasipjournals.com/content/2010/1/579341>
3. <http://www.gazecom.eu/FILES/ludw08.pdf>
4. <http://cs.haifa.ac.il/hagit/courses/ip/Lectures/Ip11_MultiscaleRepx4.pdf>
5. <http://www.ijcaonline.org/volume9/number2/pxc3871832.pdf>
6. Danescu – PI - <http://users.utcluj.ro/~rdanescu/pi_c05.pdf> - Operatii Morfologice
7. Danescu – PI - <http://users.utcluj.ro/~rdanescu/pi_c08.pdf> - Zgomotul in imagini
8. Salomie TP – L10 – Design Patterns
9. I Muntean – DISI – Curs 2 – Tehnici si sabloane de testare
10. ImageJ - <http://users.utcluj.ro/~rdanescu/pi_c08.pdf>

# Analiza si Fundamentare Teoretica

## Flux general

Fuziunea imaginilor consta din mai multi pasi importanti, fiecare fiind grupat intr-un modul separat cu eventuale submodule:

1. Citirea imaginilor DICOM
2. Fuziunea imaginilor
3. Postprocesare
4. Afisare rezultat



Un fir separat de executie reprezinta rularea procesului de masurarea calitatii a algoritmilor:

1. Incarcare imagini de test
2. Adaugarea algoritmilor care se doresc comparate
3. Calcularea metricilor de calitate
4. Formatarea si scrierea rezultatelor in fisiere



Ambele procese folosesc acelasi modul de algoritmi si metode de postprocesare. La fiecare proces, se pot alege algoritmii exacti ce vor fi folositi.

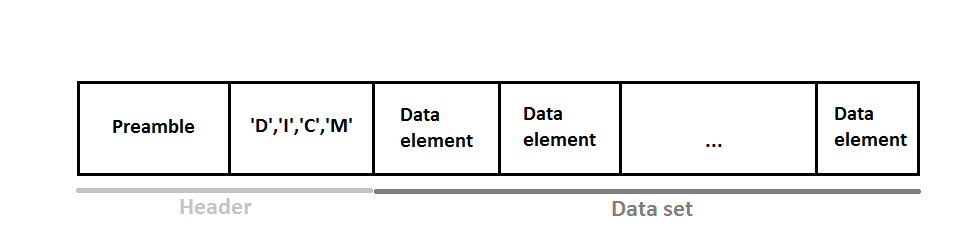
## Date de intrare si iesire

### DICOM

Tipul de date de intrare cel mai important este fisierul DICOM, avand de obicei extensia „.dcm”, dar unele informatii stocate ca o stiva de imagini pot avea extensia goala. Din aceasta cauza trebuie verificat fisierul incarcat din punctul de vedere al continutului.

In articolul [[15]](#endnote-15) putem citi ca fisierul poate fi despartit intr-un grup de antet (header) si un set de date, care reprezinta de obicei datele despre imagine. Antetul este compus dintr-un preambul de 128 de octeti urmat de un prefix DICOM de 4 bytes. Prefixul este compus din caracterele `D`, `I`, `C`, `M`, codate ca litere mari din ISO 8859 G0 Character Repertoire. Nu exista constrangerea ca preambulul sa fie structurat intr-un anumit fel, este gandit doar pentru a facilita accesul la informatiile continute de fisier.

Un set de date reprezinta un obiect care contine informatii din lumea reala. Ele pot contine valorile codificate a atributelor obiectului. Aceste atribute sunt specificate in definitiile posibile ce pot fi incluse intr-un set de date.

Un element de date este conținutul unui set de date. Este identificat unic printr-o eticheta, este ordonat crescator dupa aceasta eticheta si poate fi prezent maxim o data intr-un set de date. Sunt doua tipuri de elemente de date: standard si privat. Elementele standard au numarul de grup un numar par, iar elementele private numar impar.

Figură 4.1 Structura fisierului DICOM

### JPEG

Rezultatul procesului de fuziune va fi o imagine de format obisnuit, in prima faza afisata pe interfata grafica, si apoi putand fi salvata in format JPEG (.jpg).

In cursul [[16]](#endnote-16) gasim ca Joint Photographic Experts Group a dezvoltat acest format, care stocheaza informatia in tip bitmapped, si poate avea urmatoarele formate: jpg, jpeg, jfif, jfl. JPEG foloseste propriul algoritm de compresie, prin care se reduce semnificativ dimensiunea fizica a fisierului de imagine, dar se pierde si din calitate. Formatul suporta culori pana in 24 de biti, si poate fi folosit pe orice platforma.

Acest format de imagine va fi si intrarea la procesul de masurarea calitatii. Se foloseste JPEG in acest modul din cauza faptului sa se ofere mai multe libertate utilizatorului, in cazul in care acesta doreste sa experimenteze cu diferite set-uri de imagini de intrare, in scopul sa vada comportamentul algoritmilor de fuziune. Formatul DICOM fiind relativ greu de modificat fara software specific, s-a ales formatul JPEG care este usor de manipulat cu orice aplicatia de editare a imaginilor.

### Text

O posibila valoarea de iesire a procesului de masurare a calitatii este salvarea rezultatelor in format text, cu extensia .txt. In acest fel, informatia se poate accesa cu orice editor de text, independent de platforma, insa are dezavantajul ca nu poate fi ordonat dupa dorintele utilizatorului.

### Excel

Cea de-a doua iesire a masurarii calitatii este un fisier Excel, cu extensia .xls. In acesta se poate introduce informatia culeasa intr-o structura logica, curata si mai ales sortabila dupa diferite coloane. Acesta, la fel este independenta de platforma, insa trebuie deschis cu software special, cum ar fi Microsoft Excel pentru Windows, Libre Office Calc pentru Ubuntu/Linux si Office Web Apps pentru MacOS.

## Algoritmi

Algoritmii propusi pentru comparatia preciziei si corectitudinii a fuziunii imaginilor sunt:

* Transformata Discreta Haar Wavelet
* Piramida Laplaciana
* Metode aritmetice
  + Valoarea minima
  + Valoara maxima
  + Valoarea medie

### Transformata Discreta Haar Wavelet

#### Transformata Wavelet Discreta

Acest tip de transformare împarte un semnal sursa in componente diferite de timp-frecventa . Un semnal unidimensional de obicei se reprezintă in domeniul temporal, iar un semnal bidirecțional, cum sunt si imaginile, se reprezintă in majoritatea cazurilor in domeniul spațial. Acest al doilea tip de semnal are ca o reprezentare alternativa reprezentarea in domeniul frecvențial. Cele doua moduri de reprezentare au propriile avantaje si dezavantaje. In domeniul spațial, reprezentarea este ușor de înțeles pentru percepția umana, filtrarea se aplica direct pe datele spațiale (nu necesita transformare), însă filtrele au nuclee mari si de obicei timpul de procesare este, la fel, mai mare. In domeniul frecvențial proiectarea nucleelor de filtrare este mai ușoară, filtrarea este mai rapida însă reprezentarea in sine este non-intuitiva pentru ochiul uman si filtrările necesita o transformare in domeniul frecventei si înapoi in domeniul spațial. Trecerea din primul domeniu in celălalt si invers se poate realiza cu transformata Fourier directa si inversa .

Transformata Wavelet combina cele doua domenii (temporal si frecvențial), rezultând o aproximare atât in timp cat si in spațiu a semnalului . Se sacrifica o parte din precizia frecvențială a transformatei Fourier pentru a obține informații si despre componenta temporala a semnalului.

Exista doua metode de transformare Wavelet: transformata Wavelet continua si discreta. Prima rezulta o precizie mai mare, efectuând operații redundate pe un semnal de intrare. A doua este mai rapida, combinând perechile de date dintr-un semnal într-un mod mai eficient, însă cu pierderi minore de informație . Transformata Wavelet discreta este, din punct de vedere de procesare, mai puțin complex decât transformata Fourier, având timp de procesare O(n) fata de O(n \* logn) .

#### Transformata Haar Wavelet

Aceasta transformare este o implementare a transformatei Wavelet discrete. A fost propus de către Alfréd Haar in 1909, este cea mai simpla implementare a wavelet-urilor. Dezavantajul este ca transformata nu e una continua, deci nu este diferențiabilă (nu are derivate in oricare punct al domeniului). Aceasta proprietate poate servi si in avantajul procesării, la analiza semnalelor cu o tranziție brusca .

Transformata Haar poate fi descrisa utilizând următoarele doua funcții:

Funcția wavelet ψ(t):

Funcția de scalare ϕ(t):

Unde t reprezintă componenta temporala.

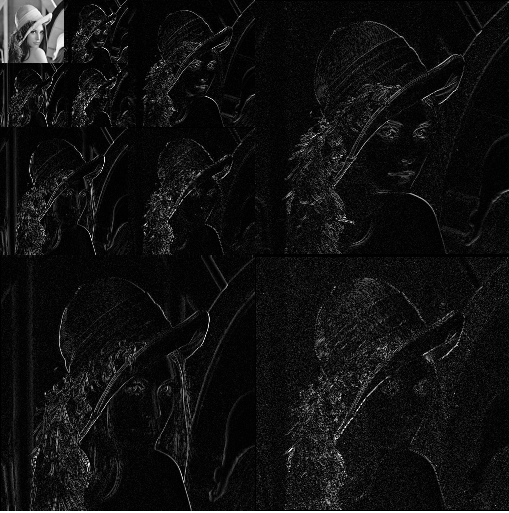
#### Implementare

Transformata Haar Wavelet unidimensionala desparte un semnal de intrare s(n) in doua semnale, j(n) si i(n), unde j(n) reprezintă semnalele de frecventa joasa, iar i(n) semnalele de frecventa înaltă. Prima data se filtrează semnalul cu un filtru trece jos si cu un filtru trece sus. Rezultatele filtrelor sunt sub eșantionate cu 2, si așa se obțin cele doua semnale j(n) si i(n). Transformata Wavelet Discreta Haar este invariant la deplasare.



Figură 4.2[[17]](#endnote-17), Haar Wavelet

Prima data se parcurge matricea de pixeli pe randuri, se calculeaza suma si diferenta elementelor consecutive. Sumele vor fi stocate intr-o jumatate a matricei, iar diferentele in cealalta jumatate. Acest procedeu se repeta pe coloane. Acesti doi pasi reprezinte o iteratie de transformare. Se pot aplica recursiv si pe matricea mai mica ramasa, simbolizand suma sumelor, in sensul ca e rezultatul sumelor de pe randuri si de pe coloane.



Figură 4.3[[18]](#endnote-18) Lena descompusa cu Haar Wavelet

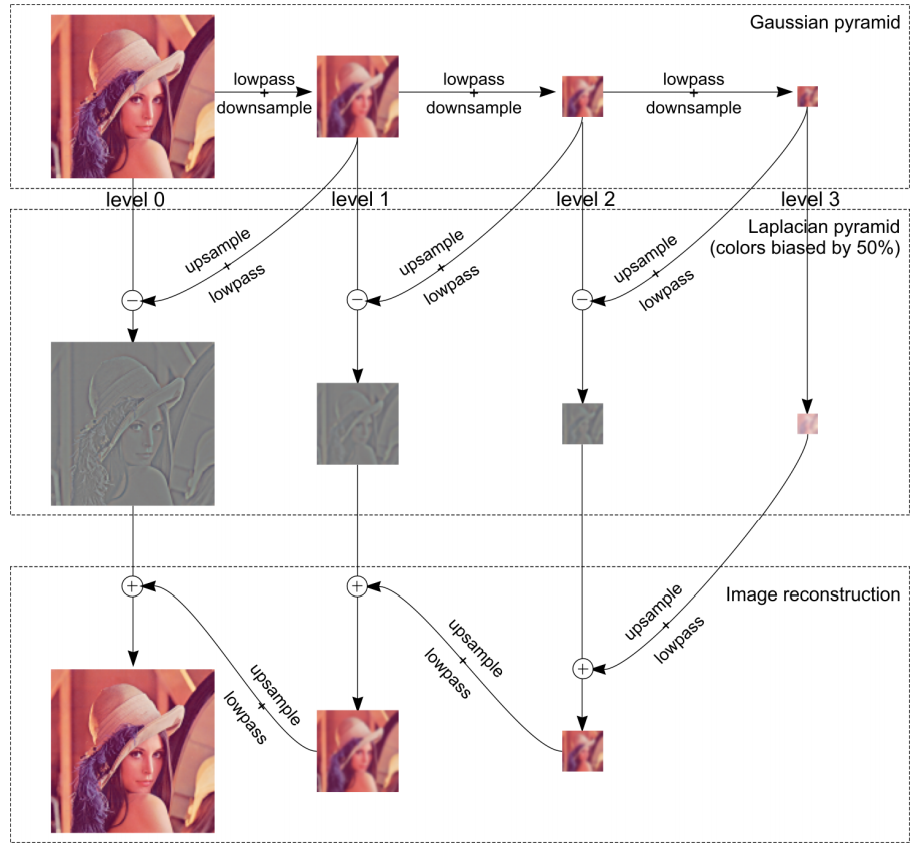
Proprietati ale transformatei Haar:

* Ortogonal
* Vectorii de baza sunt ordonate in timp
* Pierdere minima de informatie

### Piramida Laplaciana

Metoda piramidala de fuziune oferă detalii mai precise in zone de contrast mare. Fuziunea propriu zisa are loc in domeniul transformatei.

Imaginea este descompusa in imagini mai mici, prin aplicarea filtrelor trece jos si trece sus, si scalarea imaginilor rezultate. Scalarea de regula divide dimensiunile imaginii de pe nivelul precedent cu 2. Imaginea de scala cea mai mica va contine frecventele joase din imaginea sursa, iar celelalte imagini vor avea frecventele inalte. Fuziunea se face combinand informatiile de pe fiecare nivel, iar imaginea rezultat se calculeaza aplicand transformata piramidala inversa pe acesta.



Figură 4.4 [[19]](#endnote-19) Lena descompusa cu piramida Gaussiana si Laplaciana

Algoritmul de fuziune este compus din 3 parti:

1. Decompozitie: Aplicarea filtrelor trece jos si scalarea imaginilor pana ce sa ajunge la un nivel dorit
2. Fuziunea imaginilor: se combina informatiile de pe fiecare nivel folosind ori metoda aritmetica de calcularea mediei sau calcularea maximului, sau alternativ, se poate returna doar una dintre imagini, daca se constata faptul ca ar contine deja informatii mai precise decat cealalta
3. Reconstructie: Aplicarea algoritmului piramidal invers pe nivelele de imagine rezultate. Consta din urmatorii subpasi:
   1. Imaginea de pe nivelul cel mai de jos este redimensionata la 2 ori dimensiunea ei
   2. Se aplica un filtru cu matricea de convolutie transpusa decat matricea aplicata la procesul de decompozitie, folosita in filtrul trece jos
   3. Se fuzioneaza cu adunarea valorii pixelilor cu piramida de pe nivelul respectiv din decompozitie
   4. Se repeta pasul acesta imaginea rezultata fiind imaginea de intrare pentrul iteratia urmatoare

### Metode aritmetice

Doua imagini se pot fuziona si prin metode aritmetice simple, efectuand operatii matematice intre pixelii adiacenti.

Cateva metode aritmetice prezentate in aceasta lucrare, sunt metoda valorii minime, maxime si mediei intre doi pixeli. Cum sugereaza si numele metodelor, in prima se va asigna imaginii fuzionate pixelul cu intensitate mai mica dintre cele doua imagini, aici vorbindu-se despre imagini greyscale. In cazul valorii maxime, se procedeaza invers, iar in cazul valorii medii, se va lua valoarea medie a celor doi pixeli. In cazul valorii mediei pixelul rezultat va lua valoarea mediei aritmetice a pixelilor de pe pozitiile aferente din imaginile de intrare.

### Convertirea stivei de imagini in proiectie 2D

Presupunem ca avem o stiva de imagini, care reprezinta acelasi obiect la distanta sau adancime diferita. Notam dimensiunile stivei: x, y sunt latimea si lungimea imaginlior individuale, iar z este lungimea stivei, sau numarul de imagini din stiva. In acest stitem, putem trage 3 axe imaginare, x, y si z. Proiectia imaginilor se va face pe axa z. In acest procedeu, se incearca suprapunerea tuturor imaginilor, combinand valorile pixelilor in asa fel, incat sa rezulte o singura imagine ca rezultat cu informatia din toate celelalte imagini. Procedeul se poate face, de exemplu, luand valorile maxime din fiecare pixel dintre toate imaginile.

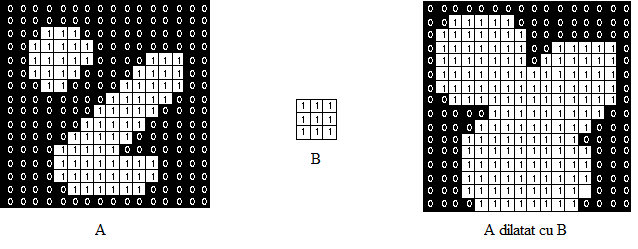
## Metode de postprocesare

### Dilatare

Din cursul [[20]](#endnote-20), aflam ca dilatarea si eroziunea sunt baza operatiilor morfologice. Daca avem 2 matrici de pixeli, A si B, putem scrie formula dilatarii binare A cu B:

Sau

Unde B – este un element structural



Figură 4.5 [[[21]](#endnote-21)], Dilatare

Conform articolului [[22]](#endnote-22), dilatarea imaginilor grayscale se poate scrie in felul urmator:

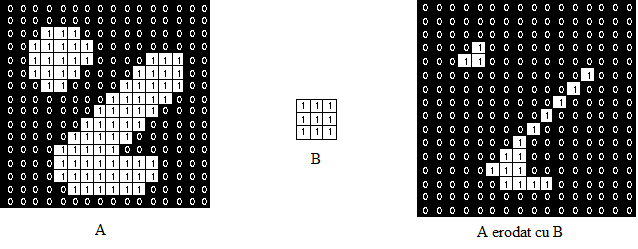
Unde:

* f(x) este functia de imagine
* b(x) este functia de dilatare
* E este spatiul Eucledian
* „sup” este supremum

Practic, prin dilatare creste suprafata totala a obiectelor binare dintr-o imagine.

### Eroziune

Tot din iii aflam ca eroziunea si dilatarea sunt duale sau altfel spus complementare. Gasim formula pentru eroziunea binara:



Figură 4.6 [[[23]](#endnote-23)], Eroziune

Conform articolului [[[24]](#endnote-24)], eroziunea imaginilor grayscale se poate scrie sub forma:

Unde B este spatiul pe care b(x) este definit si „inf” este infimum.

Practic, prin eroziune va scade aria totala a obiectelor dintr-o imagine.

### Netezire (smoothing)

Din articolul [[25]](#endnote-25), putem formula definitia procesului de netezire a imaginilor: este crearea unei aproximari in scopul de a captura sabloanele si modelele importante de date, in acelasi timp eliminand zgomotul sau alte sctructuri de dimensiuni mici, cu continut irelevant sau eronat. Se transforma semnalul de intrare intr-unul mai neted la iesire, reducand dimensiunea punctelor de zgomot, si maring dimensiunea punctelor care sunt de valori cele mai mici intre punctele adiacente.

Netezirea se poate face, de exemplu, cu un filtru Gaussian, care ofera si un efect de blur pe imagine.



Figură 4.7 Lena, imaginea originala



Figură 4.8 Lena, Gaussian blur cu radius de 5

### Metode combinate

Pentru a ajunge la rezultate mai bune, metodele prezentate mai sus se pot combina intre ele, dar trebuie tinut cont de ordinea in care acestea se executa.

De exemplu operatiunea de eroziune urmata de dilatare se numeste deschidere, si se foloseste pentru netezire de contururi, umplere goluri mici in obiecte si spargerea legaturilor slabe intre obiecte, cum scrie si in [20].

Operatiunea de dilatare urmata de eroziune este numita inchidere, si se foloseste la fel ca deschiderea, cu diferenta ultimului atribut, care acum este unirea legaturilor slabe intre obiecte.

Pentru varietate si a ajunge la mai multe rezultate posibile, se pot folosi operatiuni de dilatare urmate de netezire.

## Masurarea calitatii

Pentru a putea compara obiectiv doua imagini si analitic doua imagini diferite, trebuie sa folosim o metoda de măsurare a calității imaginii. Pentru a avea o imagine cu care sa comparam rezultatele fuziunii, vom folosi ca standard o “imagine perfecta”. Cele doua metode de masurare implementate in aceasta lucrare sunt:

* Eroarea medie patratica (Mean squared error – MSE)
* Raportul intre semnalul de varf si zgomot (Peak signal to noise ratio – PSNR)

Presupunem urmatoarele notatii:

P – imaginea perfecta

F – imaginea fuzionata

m – numarul de pixeli de pe randuri

n – numarul de pixeli de pe coloane

Pentru a putea evalua, imaginile trebuie sa aiba aceleasi dimensiuni.

Eroarea medie patratica se poate calcula cu urmatoarea formula:

Raportul intre semnalul de varf si zgomot:

Unde v este pixelul de valoare maxima.

O valoare cat mai mica reprezinta calitate mai buna in cazul primei metode de masurare, iar in un raport semnal de varf si zgomot cat mai mic semnifica la fel calitate mai buna.

Modul in care am implementat aceste masuratori este consta in mai multe module decuplabile, in acest fel aspectele care se doresc a fi masurate sunt personalizabile. Procedeul de masurare a calitatii consta in urmatorii pasi principali:

1. Se incarc grupuri de imagini de intrare.

Un grup deste compus dintr-o combinatie de trei imagini: o imagine perfecta, si doua imagini care contin informatii partiale, pe care se doreste fuziunea

1. Se incarc metodele de fuziune pe care le dorim sa analizam

Aici se poate alege numarul de metode care se vor folosi pentru fuziune, si in acest fel si pentru compararea rezultatelor.

1. Se parcurge lista de imagini de intrare, pentru fiecare grup de imagini:
   1. Se parcurge lista de metode de fuziune, si se aplica algoritmul de fuziune asupra celor doua imagini cu informatii partiale.
   2. Pentru fiecare metoda de fuziune:
      1. Se parcurg toate metodele de postprocesare, si se aplic pe imaginea rezultata din fuziune
      2. Pe aceasta imagine se vor calcula metricile de calitate, comparand-o la imaginea perfecta de la intrare
      3. Rezultatele se pastrec ca grupuri de informatie de iesire in urmatorul format:

ImagineRezultat, MSE, PSNR

Pentru afisarea rezultatelor am implementat un alt modul, care ofera utilizatorului posibilitatea sa aleaga prin ce format vrea sa se salveze rezultatele masuratorilor de calitate. La momentul de fata, se pot alege intre doua feluri de scriere a rezultatelor: in format text (.txt) sau in format Excel (.xls). Ambele se doresc sa salveze rezultatele intr-o forma cat mai lizibila.

Avantajul metodei al doilea consta in faptul ca rezultatele se pot ordona in functie de coloana, iar pentru fiecare grup de imagini se creaza o pagina noua (sheet). Dezavantajul este ca nu toate sistemele au program de citire a fisierelor .xls.

# Proiectare de Detaliu si Implementare

## Prezentare generala

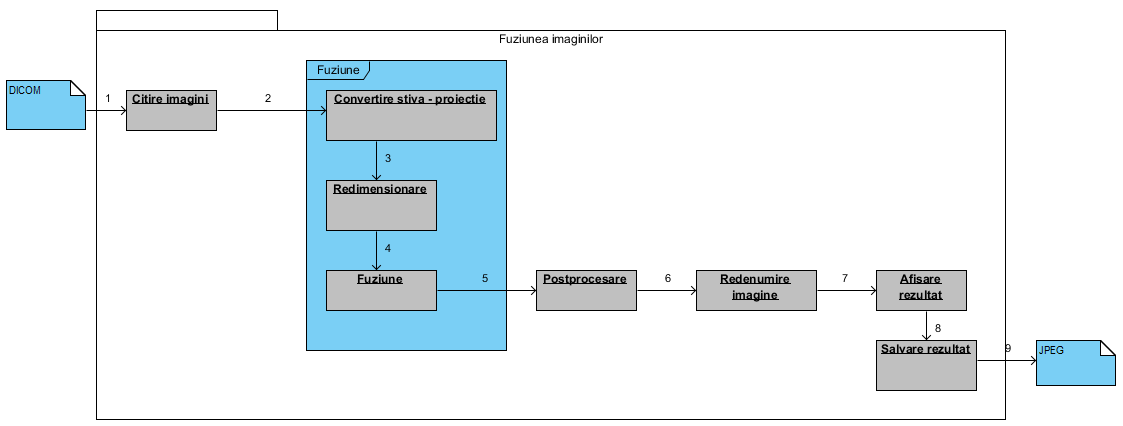
In proiectarea sistemului am urmarit despartirea functionalitatilor pe module, care la implementare vor fi reprezentate de pachete de cod sursa. Modulele au un rol bine definit in fluxul de date a aplicatiei, ele fiind independente una de alta, dar functionarea corecta nu poate fi obtinuta fara acestea sa coopereze si sa se sincronizeze in procesele sistemului. Un controller principal este responsabil de coordonarea modulelor, si definirea ordinea pasilor care trebuie urmate pentru a obtine rezultatele dorite si acestea sa ajunga la utilizator, fie prin interfata grafica sau prin salvarea lor pe hard disk.

Modulele principale si submodulele aferente sunt:

* Cititorul de imagini DICOM
* Modulul de fuziune
  + Convertitorul din stiva de imagini in proiectie
  + Redimensionarea imaginilor
  + Fuziunea cu algoritmul si parametrii alesi
* Postprocesare cu metoda aleasa
* Redenumirea imaginilor rezultat
* Afisare rezultat
* Salvare rezultat ca imagine JPEG

Pentru fluxul procesului de masurarea calitatii modulele sunt prezentate in capitolul anterior. Descrierea detaliata a modulelor se poate gasi in cele ce urmeaza.

## Arhitectura generala a sistemului



Figură 5.1



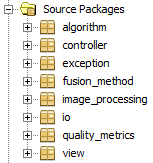
Figură 5.2

Fluxurile generale ale informatiei sunt dictate de cerintele functionale ale sistemului, prezentate anterior, si diagrama de use case.



Figură 5.3 Diagrama use-case a aplicatiei

Despartirea modulelor in pachete de cod sursa a fost efectuat conform bunelor practici ale programarii in limbajul Java. Structura arata in felul urmator:



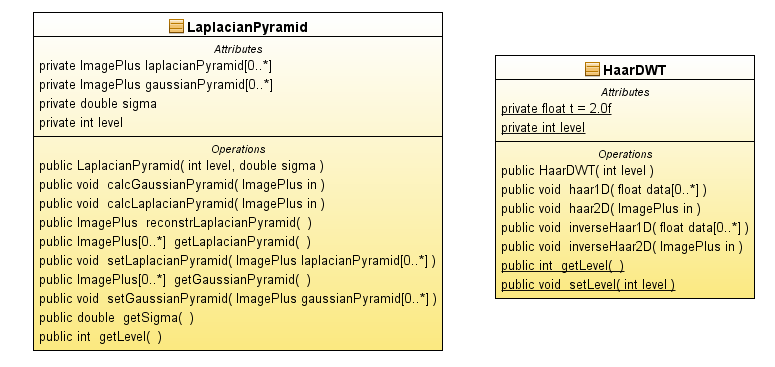
Figură 5.4 Structura de pachete a aplcatiei

Proiectarea aplicației a fost făcută folosind design pattern-ul MVC (Model View Controller), prin acest most separand clar nivelele de prezentare, model si control.

## Descrierea modulelor

### Modulul de algoritmi

Modulul poate cel mai important este modulul in care se afla algoritmii care stau la baza metodelor de fuziune. In acesta sunt definite cei doi algoritmi, Transformata Discreta Wavelet Haar si Piramida Laplaciana. Cei doi fac categorie din clase diferite de algoritmi de fuziune, primul fiind de tip Wavelet, iar cel de-al doilea fiind de tip piramidal. Implementarea lor este in pachetul „algorithm”, in clasele „HaarDWT” respectiv „LaplacianPyramid”. Se pot observa din numele metodelor, ca acesti algoritmi presupun si o modalitate de a inversa procesul de transformare, altfel zis, reproducerea semnalului original de intrare dupa transformarea acestuia.



Figură 5.5 Diagrama de clasa a pachetului „algorithm”

#### Transformata Wavelet Discreta Haar

Transformata Wavelet Discreta Haar directa 1D:

int h = data.length / 2;

for (int i = 0; i < h; i++) {

int k = i \* 2;

temp[i] = (data[k] + data[k + 1]) / t;

temp[i + h] = (data[k] - data[k + 1]) / t;

}

* „data” este semnalul de intrare (imaginea reprezentata ca un vector), de exemplu un rand din imagine sau o coloana.
* „h” reprezinta mijlocul vectorului de intrare.
* „i” itereaza de la inceputul vectorului pana la „h”
* „k” este folosit sa luam elementele din doua in doua pozitii
* „temp” este vectorul care va contine transformata wavelet la iesire
* „t” este factorul de despartire, care e de obicei 2

\*diagrama

Transformata 1D inversa funcioneaza similar, cu diferenta construirii vectorului de iesire:

for (int i = 0; i < h; i++) {

int k = i \* 2;

temp[k] = data[i] + data[i + h];

temp[k + 1] = data[i] - data[i + h]

}

Transformatele 2D folosesc aceste metode, aplicandu-le pe coloane si randuri. Transformata Wawelet Haar este implementata ca fiind multirezolutional, se poate preciza nivelul pana la care se vor aplica transformatele 1D:

int levCols = cols / lev;

int levRows = rows / lev;

* „levCols” este nivelul coloanelor
* „cols” este numarul total al coloanelor
* „lev” este nivelul dorit de rezolutie
* Analog pentru randuri, „levRows” si „rows”

Folosind aceste valori, se itereaza prima data peste randuri, calculand transformatele Haar 1D:

row = new float[levRows];

for (int i = 0; i < levRows; i++) {

for (int j = 0; j < row.length; j++) {

row[j] = data[j][i];

}

haar1D(row);

for (int j = 0; j < row.length; j++) {

data[j][i] = row[j];

}

}

* Se creaza un nou rand „row” de dimensiunea randurilor pentru nivelul de rezolutie aferent, „levRows”
* Se itereaza in patratul levCols \* levRows din datele de intrare „data”
* Se extrag pixelii de pe pozitiile aferente din „data”, se pun in „row”
* Se aplica transformata Haar Wavelet directa pe „row”
* Se scrie rezultatul obtinut in randul din care s-a citit, inapoi in datele de intrare

Similar se aplica procedeul si pentru coloane. La sfarsitul procedeului vom avea in „data” transformata Haar Wavelet Discreta.

Analog se face si transformata inversa. Codul de mai jos exemplifica diferentele:

for (int i = 0; i < levRows; i++) {

for (int j = 0; j < row.length; j++) {

row[j] = data[j][i];

}

inverseHaar1D(row);

for (int j = 0; j < row.length; j++) {

data[j][i] = row[j];

}

}

#### Piramida Laplaciana

Calcularea piramidei Laplaciane consta in doi pasi importanti:

1. Calcularea Piramidei Gaussiene
2. Calcularea Piramidei Laplaciene

Cum sugereaza si numele, semnalul de intrare este transformat intr-o reprezentare de forma unei piramide, adica o sa avem o list de imagini dintr-una de intrare de lungimea nivelului dorit de transformare. Aceasta lista o sa fie reprezentata ca un ArrayList<ImagePlus>.

Procesul de calculare a piramidei Gaussiene est urmatorul:

1. Se citeste o imagine de intrare
2. Se adauga in lista de imagini ArrayList<ImagePlus>
3. Pentru fiecare nivel de rezolutie
   1. Se aplica un filtru trece jos, sau filtru Gaussian pe imaginea precedenta, asigurand un efect de blur
   2. Se redimensioneaza imaginea precedenta divizand dimensiunile cu 2
   3. Se adauga in lista de imagini pe pozitia nivelului curent

iP.blurGaussian(sigma);

ImageProcessor iPResized = iP.resize(width / 2);

Procesul de calculare a piramidei Laplaciene:

1. Se ia piramida Gaussiana calculata anterior
2. Se incepe de la nivelul cel mai inferior a piramidei
3. Se extrage imaginea de pe nivelul curent
4. Se maresc dimensiunile cu un factor de 2
5. Se aplica un filtru trece-jos, sau blur Gaussian
6. Se calculeaza diferentele de pixeli intre imaginea pe care s-au efectuat aceste operatii si imaginea precedenta din piramida Gaussiana
7. Se adauga imaginea obtinuta in piramida Laplaciana, pe nivelul aferent

for (int i = level - 1; i > 0; i--) {

ImagePlus gaussian = gaussianPyramid.get(i).duplicate();

ImageProcessor processor = gaussian.getProcessor();

// size \*= 2

processor.setInterpolationMethod(ImageProcessor.BICUBIC);

ImageProcessor processorResized = processor.resize(gaussian.getWidth() \* 2);

gaussian = new ImagePlus("GaussianResized " + i, processorResized);

// low pass filter

processorResized.blurGaussian(sigma);

//difference between current image and previous

ImagePlus current = gaussianPyramid.get(i - 1);

ImageCalculator calculator = new ImageCalculator();

ImagePlus difference = calculator.run("Subtract create", current, gaussian);

difference.setTitle("Laplacian " + (i - 1));

// store laplacian image

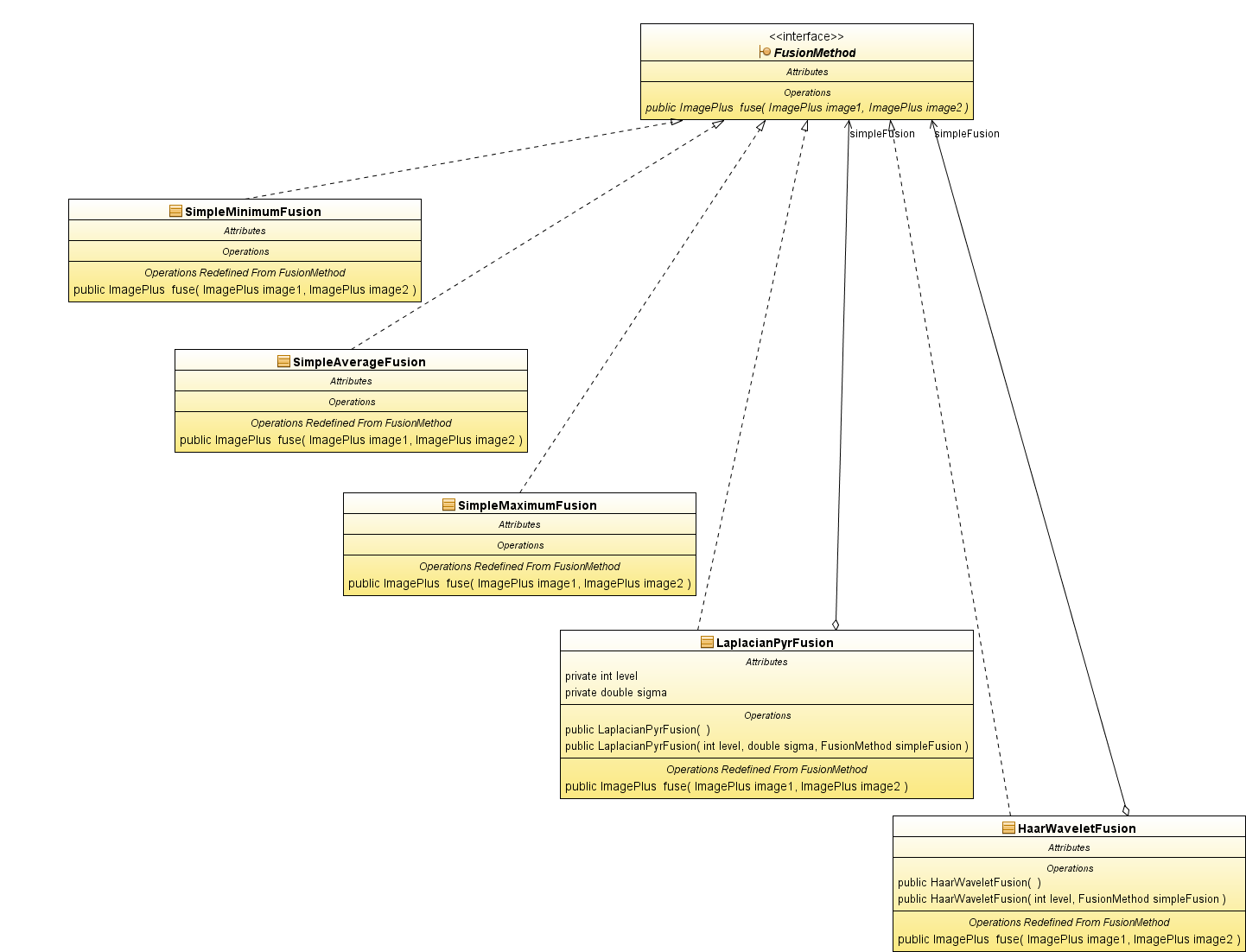
laplacianPyramid.add(difference);

}

Reconstructia imaginii originale din piramida Laplaciana este similara cu algoritmul prezentat mai inainte, cu diferenta ca se adauga valorile pixelilor la fiecare pas la o imagine ImagePlus. Suma totala va reprezenta imaginea retransformata.

### Modulul de metode de fuziune

Acest modul contine metodele propriu-zise de fuzionare. Cateva dintre clase folosesc ca algoritm de baza algoritmii prezentati mai sus in modulul precedent. Metodele de fuziune trebuie sa implementeze interfata „FusionMethod”, si metoda acestuia „public ImagePlus fuse(ImagePlus image1, ImagePlus image2)”. Aceasta metoda va contine toata logica din spate a fuzionarii.



Figură 5.6 Diagram de clasa a pachetului „fusion\_method”

Clasa „ImagePlus” din pachetul „ij” (ImageJ), reprezinta o imagine cu metode de procesare si filtrare incluse [[[26]](#endnote-26)]: contine un procesor de imagine de tip ImageProcessor, sau o stiva de imagini 3D, 4D sau 5D, de tip ImageStack. In afara de aceste obiecte, ImagePlus poate contine si metadate, cum ar fi calibrarea spatiala, numele fisierului sau a directorului din care a fost citita.

Metodele de fuziune mai complexe, cum sunt clasele HaarDWT si LaplacianPyramid, contin o referinta la un obiect FusionMethod. Acesta este folosit la fuziunea propriu-zisa a rezultatelor obtinute din aplicarea algoritmilor pe imaginile de intrare. Cum acestea transforma semnalul de intrare, procedeul general de fuziune in cazul lor este:

1. Transformarea imaginilor folosind algoritmul ales
2. Fuziunea rezultatelor cu o metoda de fuziune aritmetica
3. Transformarea rezultatului fuziunii inapoi in domeniul original al imaginii

Exemplu de implementare a metodei „fuse” in clasa „HaarWaveletFusion”:

public ImagePlus fuse(ImagePlus image1, ImagePlus image2) {

ImagePlus haarImage1 = image1.duplicate();

haarDwt.haar2D(haarImage1);

ImagePlus haarImage2 = image2.duplicate();

haarDwt.haar2D(haarImage2);

// Fusion method used to fuse the two haar images

ImagePlus result = simpleFusion.fuse(haarImage1, haarImage2);

haarDwt.inverseHaar2D(result);

result.setTitle("Haar\_" + HaarDWT.getLevel() + " " + image1.getShortTitle() + " + " + image2.getShortTitle());

return result;

}

* Se duplica imaginile de intrare, ca sa nu se faca modificari direct pe ele
* „haarDwt” este algoritmul Haar de tipul „HaarDWT”
* Se calculeaza transformatele Haar pentru fiecare imagine
* Se face fuziunea intre transformatele Haar cu o metoda simpla (aritmetica)
* Se aplica transformata inversa pe rezultatul fuziunii
* Se redenumeste imaginea rezultat si se returneaza

Analog functioneaza si metoda piramidei Laplaciene, insa acolo se face fuziunea pe toate elementele piramidelor separat, rezultatul fiind o noua piramida. Pe aceasta se va aplica transformata inversa:

for (int i = 0; i < pyramid1.size(); i++) {

result.add(simpleFusion.fuse(pyramid1.get(i), pyramid2.get(i)));

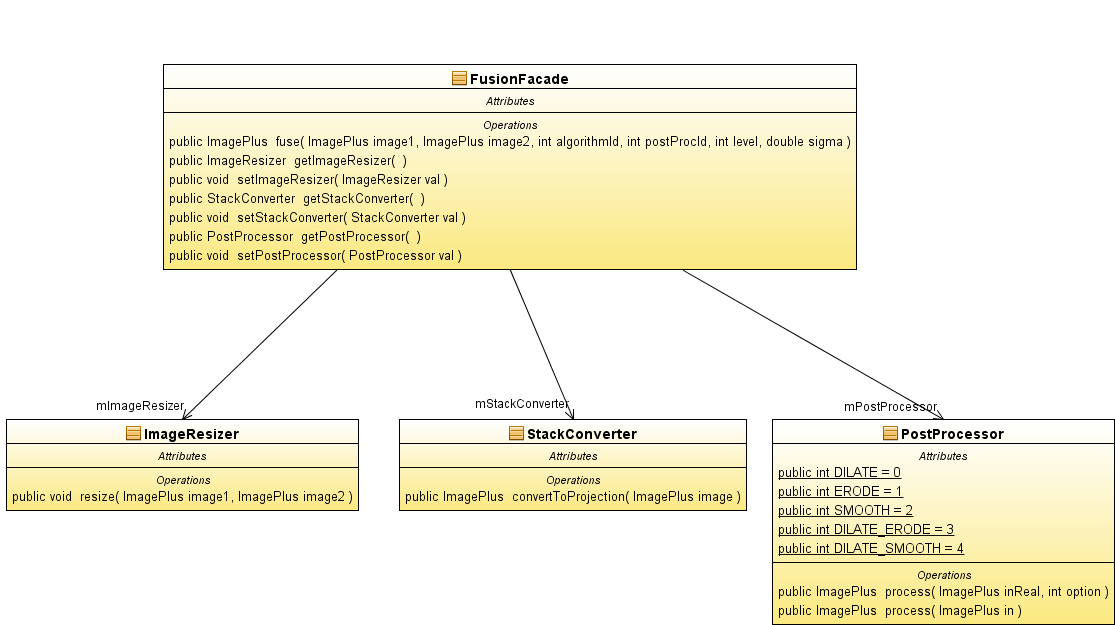
}

Iar imaginea rezultat se extrage cu:

ImagePlus resultImage = pyramidCalculator.reconstrLaplacianPyramid();

### Modulul de procesare a imaginilor

In acest modul sunt definite operatii generale de procesarea imaginilor, care sunt importante pentru aplicatia de fata, insa nu esentiale in functionarea teoretica. Practic, ele se folosesc pentru a ajunge la un rezultat cat mai bun dupa aplicarea fuziunii.



Figură 5.7 Diagrama de clase a pachetului „image\_processing”

ImageResizer este folosit doar in cazul in care dimensiunile imaginilor de intrare difera. In cazul acesta, se vor redimensiona la media dimensiunilor lor originale, in acest mod se minimizeaza cantitatea de informatii pierdute si cantitatea de informatii adaugate prin interpolarea pixelilor (in cazul in care se mareste dimensiunea unei imagini).

Unde:

* R este imaginea rezultat
* A si B sunt imaginile de intrare
* W este latimea imaginilor
* H este inaltimea imaginilor

Clasa StackConverter converteste stivele de imagini in proiectia lor pe axa verticala imaginara Z. Pentru acest procedeu ne vom folosi de clasa deja implementata ca un plugin ImageJ, ZProjector, din pachetul „ij.plugin”.

Un exemplu de cod sursa sugereaza functionarea acestei clase:

ZProjector projector = new ZProjector(image);

projector.setMethod(ZProjector.MAX\_METHOD);

projector.doProjection();

Clasa PostProcessor incapsuleaza logica de postprocesare a imaginilor. Pentru aceasta operatiune vom folosi PostProcessor-ul din imaginea de intrare de tip ImagePlus. Exemplu de postprocesarea aplicand dilatarea:

ImageProcessor processor = in.getProcessor();

processor.dilate();

Aici „in” este imaginea de intrare de tip ImagePlus.

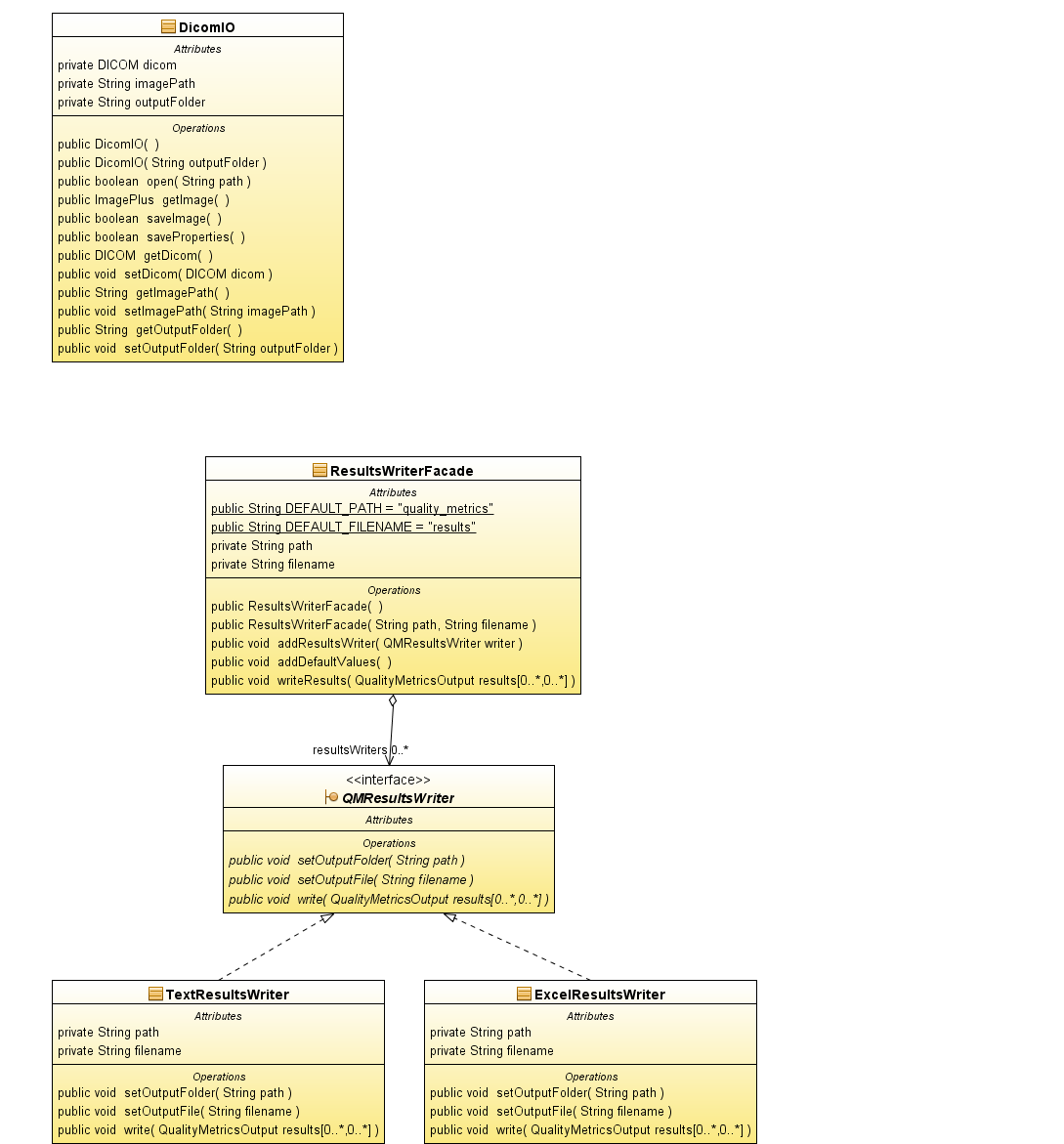
Cu un switch() se verifica optiunea aleasa de utilizator si se alege metoda de postprocesare aferenta pentru aceasta:

in.setTitle(inReal.getTitle() + ">dilated");

Toata logica de procesarea a imaginlior este grupata intr-o facada, FusionFacade, prin care se ascunde logica de implementare, si se ofera acces usor la operatiile importante. Aceasta fatada delega responsabilitatile la obiectele enumerate mai sus.

### Modulul de citirea si scriere data

Acest modul este responsabil de citirea datelor din fisiere si scrierea rezultatelor in forma de fisiere noi.



Figură 5.8 Diagrama de clase a pachetului „io”

Prin intermediul acestui modul putem deschide fisierele DICOM, extragand informatiile din ele in format ImagePlus. Acesta foloseste clasa „DICOM” din ImageJ, care ne ofera o serie de metode deja implementate, cum ar fi deschiderea unui fisier dicom, citirea dimensiunii stivei de imagini continute in fisier, extragerea informatiilor din set-ul de date si citirea imaginii propriu-zise din fisier.

Exemple de operatii:

* Initializare obiect DICOM

private DICOM dicom = new DICOM();

* Deschidere fisier DICOM cu calea „path”

dicom.open(path);

* Citirea latimii imaginii continute in fisierul DICOM

dicom.getWidth();

* Citirea dimensiunii stivei de imagini

dicom.getStackSize();

* Citirea titlului scurt al fisierului deshis (fara calea absoluta)

dicom.getShortTitle();

* Extragerea imaginii din DICOM

dicom.getImage();

* Extragerea stivei de imagini

dicom.getImageStack();

Al doilea format important de iesire a sistemului sunt rezultatele masurarii calitatii. Acestea se pot scrie fie in fisier text, fie in fisier Excel. Pentru a formata si persista fisierele Excel, am folosit framework-ul JExcel API, care este usor de integrat si ofera metode folositoare pentru salvarea informatiilor in format lizibil si structurat logic. Exemplu de folosire JExcel:

* Importarea clasei „jxl.Workbook”
* Creare „caiet” excel

WritableWorkbook workbook = Workbook.createWorkbook(new File(this.path + "/" + this.filename + ".xls"));

* Creare paginilor in interiorul cartii excel

WritableSheet sheet = workbook.createSheet("Results " + sheetIndex, sheetIndex);

Aici sheetIndex este un numar intreg, care reprezinta numarul foii din carte.

* Crearea unui camp nou care contine String

Label label = new Label(0, rowIndex, output.getResultImage().getTitle());

* Adaugarea campului la foaia creata mai inainte

sheet.addCell(label);

* Scrierea informatiilor adaugate in caiet

workbook.write();

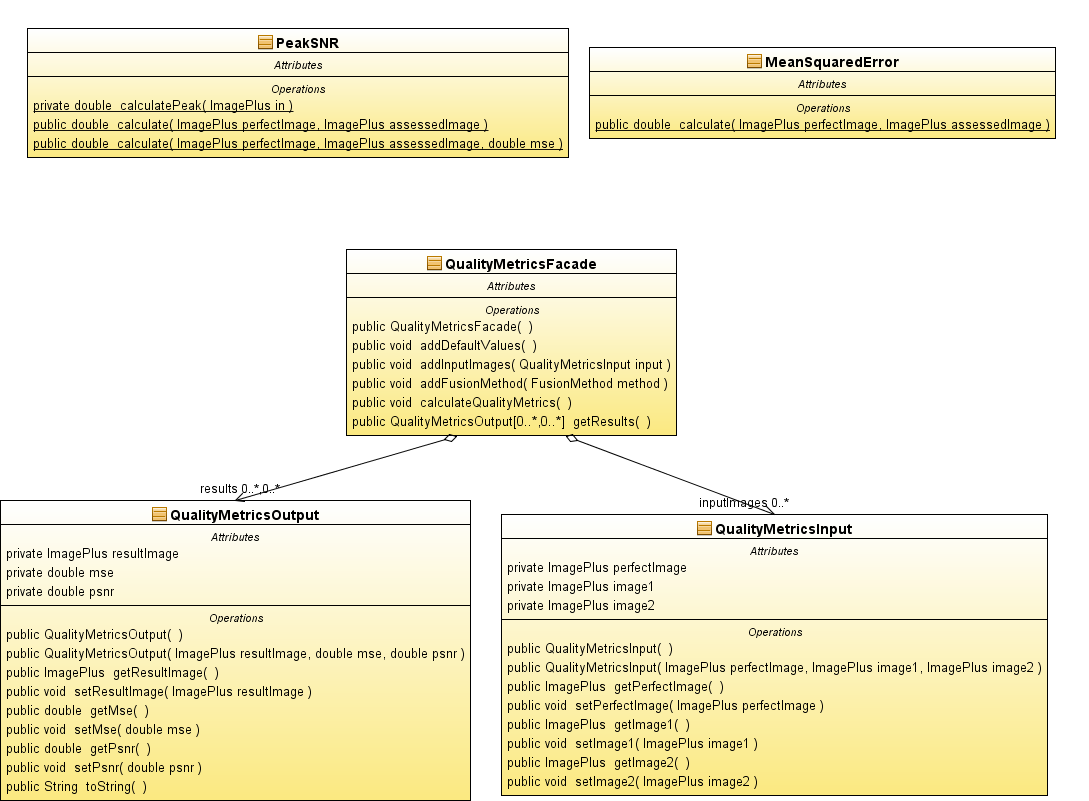
* Inchiderea caietului de lucru

workbook.close();

### Modulul de metrici de calitate

Pentru calcularea metricilor de calitate am folosit doua masuratori:

* Eroarea medie patratica (Mean squared error – MSE)
* Raportul intre semnalul de varf si zgomot (Peak signal to noise ratio – PSNR)



Figură 5.9 Diagrama de clase pentru pachetul „quality\_metrics”

Functionarea lor teoretica si formula sunt prezentate in Capitolul 3. Clasa MeanSquaredError contine metoda de calculare a eroarei medii patratice si este implementata in felul urmator: pentru fiecare pixel al imaginii care se testeaza, se calculeaza diferenta de intensitate intre pixelul imaginii de intrare cu pixelul unei imagini „perfecte”, si numarul acesta se ridica la patrat. Se adun toate valorile calculate si se divid la sfarsit cu numarul total al pixelilor din imagini.

for (int i = 0; i < perfectImage.getHeight(); i++) {

for (int j = 0; j < perfectImage.getWidth(); j++) {

error += Math.pow(perfectProc.get(i, j) - assessedProc.get(i, j), 2);

}

}

return error / (perfectImage.getWidth() \* perfectImage.getHeight());

Unde perfectProc este ImageProcessor-ul imaginii perfecte, iar assessedProc este ImageProcessor-ul imaginii supuse testului.

Clasa PeakSNR contine metodele pentru calcularea raportului intre semnalul de varf si zgomot. In prima faza se calculeaza intensitatea maxima a pixelilor din imaginea masurata, dupa care se calculeaza MSE pentru imagine si se calculeaza PSNR cu urmatoarea formula:

Codul care calculeaza acest numar:

double mse = MeanSquaredError.calculate(perfectImage, assessedImage);

double peak = calculatePeak(assessedImage);

double psnr = 10 \* Math.log10(Math.pow(peak, 2) / mse);

Aceste doua modele reprezinta precizia, sau mai bine zis similaritatea imaginii supuse testului cu o imagine „perfecta”. Eroarea medie patratica mai mica reprezinta similaritate mai mare, iar raportul intre semnalul de varf si zgomot mai mare inseamna la fel, calitate mai buna.

Pentru rularea procesului da masurarea calitatii, modulul primeste la intrare obiecte multiple de tip QualityMetricsInput, care incapsuleaza trei imagini ImagePlus: imaginea perfecta, si cele doua imagini care urmeaza sa fie fuzionate si rezultatul lor comparat cu prima imagine.

La iesire, modulul produce un obiect de tipul QualityMetricsOutput, care va contine rezultatele procesului de masurarea calitatii: imaginea rezultat a procesului de fuziune, si cele doua valori de calitate, MSE si PSNR.

Toata logica de calcularea calitatii este incapsulata intr-o fatada, QualityMetricsFacade, care ofera acces usor la functionalitati, si in acelasi timp ascunde implementarea metodelor.

### Modulul de vizualizare si interfata grafica

Acest modul contine interfata grafica a aplicatiei. Se pot gasi doua clase in pachetul acesta: MainForm si MessageDialog.



Figură 5.10 Diagrama de clase pentru pachetul „view”

MainForm contine elementele vizuale din Java Swing prin intermediul carora utilizatorul poate interactiona cu aplicatia. Interfata e proiectata in asa fel incat sa fie intuibila si usor de folosita. Mai multe detalii despre interfata grafica si cum se poate interactiona cu aplicatia se pot gasi in Capitolul 7.

Clasa MessageDialog este folosita la afisarea eventualelor mesaje, fie de eroare sau avertizare. Este proiectata in asa fel incat tipul si titlul mesajului poate fi precizat de catre un parametru String:

if (type.toLowerCase().equals("warning")) {

messageType = JOptionPane.WARNING\_MESSAGE;

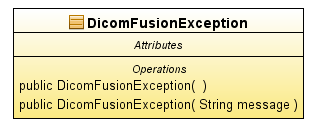
} else if (type.toLowerCase().equals("error")) {

messageType = JOptionPane.ERROR\_MESSAGE;

}

### Modulul exceptie

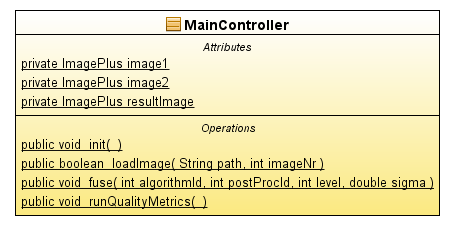
Aplicatia are definit si un modul propriu de exceptie. Aceata modalitate este preluata din bunele practici ale limbajului Java. Exceptia este aruncata la nivel de procesare si este prinsa in controller, care ii afiseaza mesajul in interfata grafica.



Figură 5.11 Diagrama clasei „Exception”

### Modulul controller

Modulul de controller este responsabil de fluxul corect al datelor si operatilor. El comunica cu stratul de vizualizare si modelele care contin logica de procesare si algoritmii.

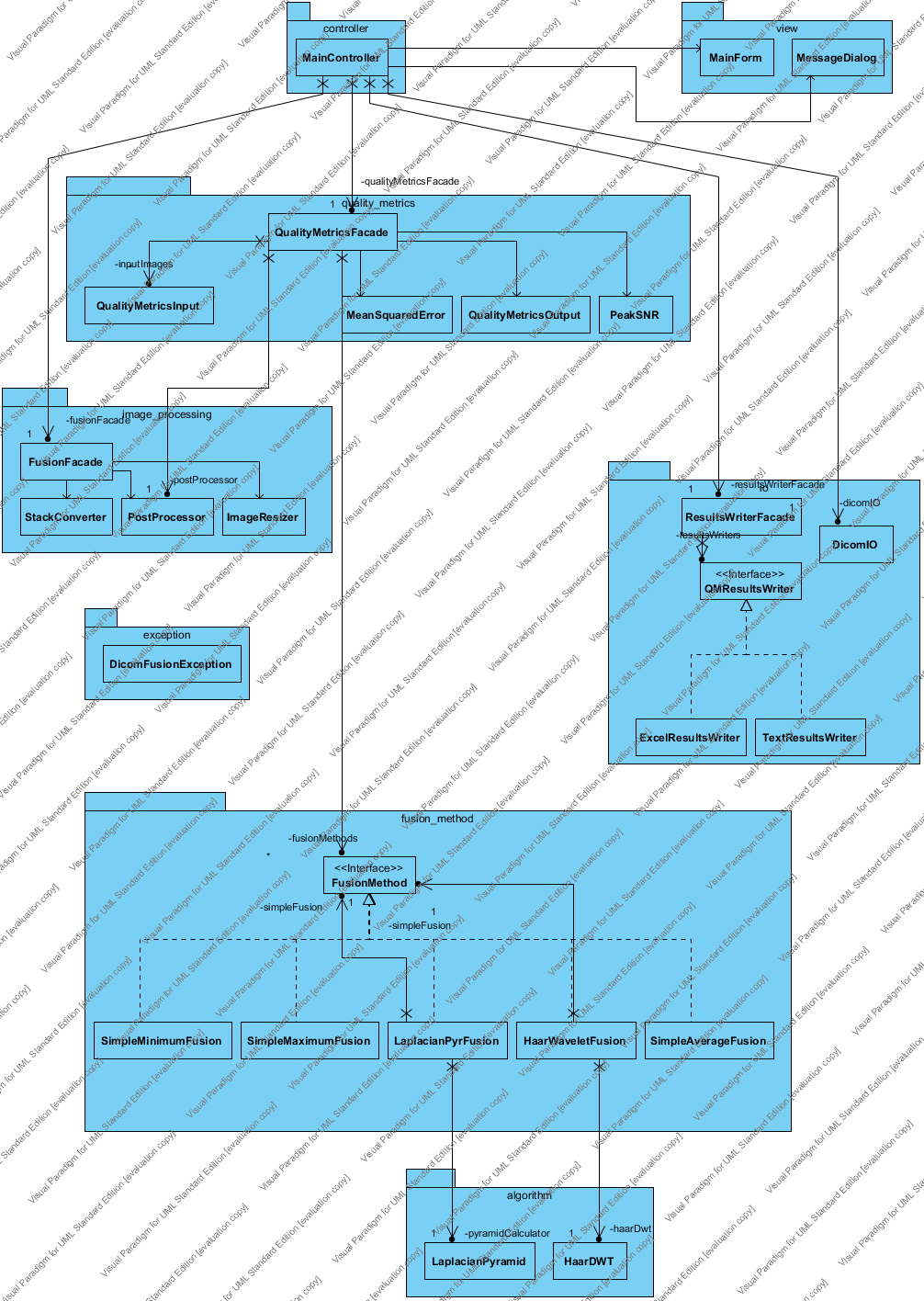


Figură 5.12 Diagrama clasei „MainController”

MainController-ul foloseste printre altele fatadele definite in celelalte module. MainController este la fel responsabil sa proceseze datele si stimulii primiti de la utilizator, sa le „inteleaga” si sa transporte cerintele mai departe la modulele de algoritmi. Pe langa asta, controller-ul intercepteaza mesajele de exceptie si le afiseaza utilizatorului prin interfata grafica.

## Vedere in ansamblu al sistemului si operatiile importante

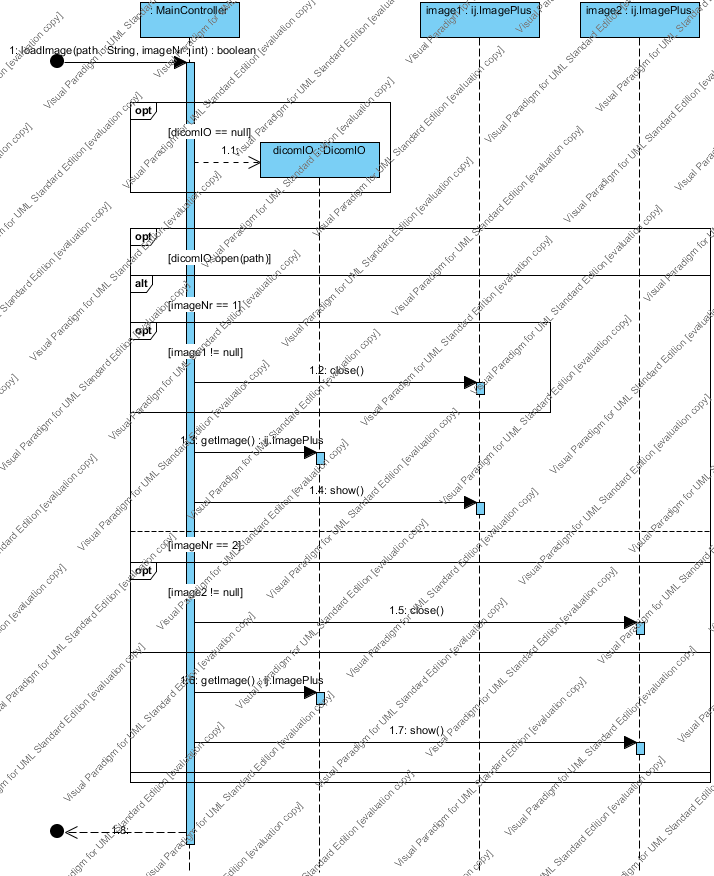
Mai jos este prezentata diagrama de clase generala pentru intreaga aplicatie. Se pot observa legaturile intre clase si interactiunea partii de interfata grafica cu logica aplicatiei.



Figură 5.13 Vedere in ansamblu al aplicatiei

Mai jos este prezentata diagram de secventa pentru deschiderea fisierelor DICOM. Pasii principali sunt:

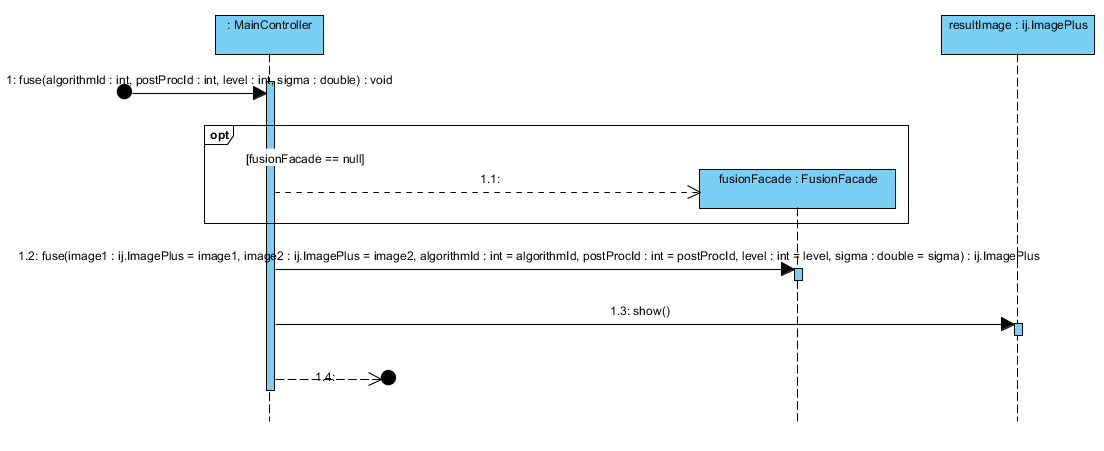
* MainController primeste comanda de la interfata grafica sa citeasca un fisier
* Apeleaza metoda din dicomIO, cu care sa deschida fisierul
* Dupa numarul transmis ca parametru, va stie care imagine a fost citita



Figură 5.14 Diagrama de secventa a operatiei de deschidere fiserelor DICOM

Pasii principali a procesului de fuziune:

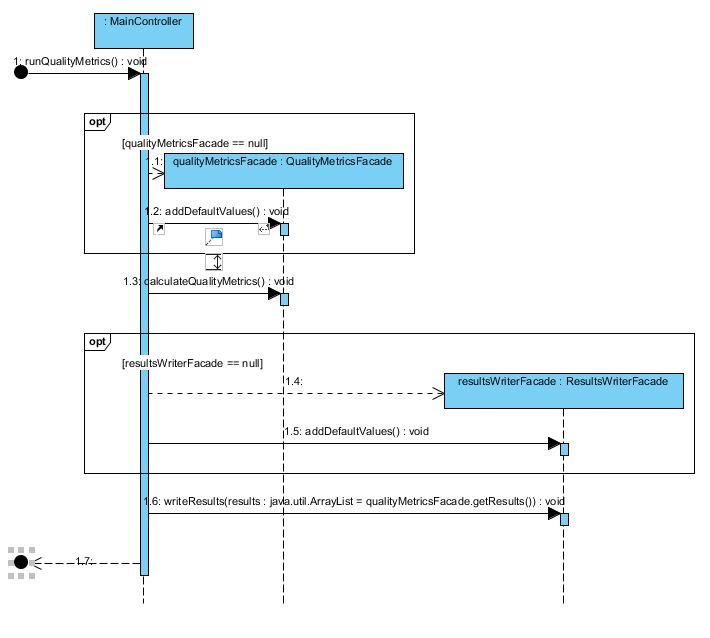
* MainController primeste comanda de la interfata grafica
* Apeleaza metoda de fuziune din fusionFacade
* Acesta incapsuleaza logica de fuziune
* Va returna o imagine ImagePlus ca rezultat



Figură 5.15 Diagrama de secventa a operatiei de fuziune, adancime de 3 nivele

Pasii principali a procesului de masurarea calitatii:

* MainController primeste comanda de la interfata grafica
* Apeleaza metoda de calculare din qualityMetricsFacade
* Cu rezultatul obtinut de la acesta, apeleaza resultsWriterFacade
* Se salveaza rezultatele in functie de formatul ales



Figură 5.16 Diagrama de secventa a operatiei de rularea metricilor de calitate

# Testare și Validare

## Teste de performanta

Sistemul pe care s-au rulat testele are urmatoarele caracteristici:

* Procesor: Intel® Core™ i5 CPU M460 @ 2.53 GHz, 2 Cores, 4 Logical Processors
* Memorie RAM instalata: 8GB, 1333 MHz
* Sistem de operare: Microsoft Windows 7 Ultimate
* Arhitectura sistem: x64

### Procesul de fuziune

Procesul de fuziune se masoara in milisecunde pentru fiecare combinatie de algoritm si metoda de postprocesare. Mai jos aveti un table cu valorile obtinute.

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm fuziune | Postprocesare | Parametri | Rezultat [ms] |
| Minim | - | - | 3 |
| Medie | - | - | 5 |
| Medie | Dilatare | - | 9 |
| Medie | Eroziune | - | 7 |
| Medie | Netezire | - | 7 |
| Medie | Dilatare + Eroziune | - | 11 |
| Medie | Dilatare + Netezire |  | 10 |
| Maxim | - | - | 4 |
| Laplacian | - | Nivel=1, sigma=3 | 5 |
| Laplacian | - | Nivel=2, sigma=3 | 56 |
| Laplacian | - | Nivel=3, sigma=1 | 69 |
| Laplacian | - | Nivel=3, sigma=2 | 66 |
| Laplacian | - | Nivel=3, sigma=3 | 67 |
| Laplacian | Dilatare | Nivel=3, sigma=3 | 74 |
| Laplacian | Eroziune | Nivel=3, sigma=3 | 71 |
| Laplacian | Netezire | Nivel=3, sigma=3 | 71 |
| Laplacian | Dilatare + Eroziune | Nivel=3, sigma=3 | 74 |
| Laplacian | Dilatare + Netezire | Nivel=3, sigma=3 | 74 |
| Laplacian | - | Nivel=3, sigma=4 | 85 |
| Laplacian | - | Nivel=4, sigma=3 | 76 |
| Laplacian | - | Nivel=5, sigma=3 | 81 |
| Haar | - | Nivel=1 | 12 |
| Haar | - | Nivel=2 | 12 |
| Haar | - | Nivel=3 | 13 |
| Haar | Dilatare | Nivel=3 | 16 |
| Haar | Eroziune | Nivel=3 | 16 |
| Haar | Netezire | Nivel=3 | 15 |
| Haar | Dilatare + Eroziune | Nivel=3 | 19 |
| Haar | Dilatare + Netezire | Nivel=3 | 19 |
| Haar | - | Nivel=4 | 13 |
| Haar | - | Nivel=5 | 13 |
| Haar | - | Nivel=6 | 13 |
| Haar | - | Nivel=7 | 13 |

Din aceste valori, putem observa ca metodele aritmetice sunt cele mai rapide, care este usor de inteles, pentru ca ele nu necesita operatii complexe la fuziune. Referitor la metodele de postprocesare, am aflat ca in general netezirea si eroziunea sunt cele mai rapide, urmate de dilatare, apoi de operatii combinate, dilatare cu eroziune (inchidere) si dilatare cu netezire.

La algoritmii multirezolutionali, timpul de procesare creste in general cu cresterea nivelului de rezolutie, mai semnificativ in cazul piramidei Laplaciene, mai putin observabil in cazul transformatei Haar.

### Procesul de masurarea calitatii

Procesul de masurare a calitatii nu poate fi parametrizat, deci cu multiple rulari incercam sa ajungem la o medie de milisecunde de procesare.

|  |  |
| --- | --- |
| Numarul rularii | Durata procesului [ms] |
| 1 | 9246 |
| 2 | 8507 |
| 3 | 8422 |
| 4 | 8448 |
| 5 | 8481 |

Deci in medie, procesul dureaza 8620 de milisecunde, sau aproximativ 8.5 de secune. De mentinut ca procesul nu include si salvarea rezultatelor in fisiere. El ruleaza cele 5 metode, fiecare cu cele 5 moduri de postprocesare, deci in total 5 \* 5 = 25 de rulari pentru fiecare set de imagini de intrare. Ca si parametru predefinit, avem 3 seturi de imagini de intrare, deci ne rezulta 3 \* 25 = 75 de rulari de fuziune in total.

## Metrici de calitate si comparatia algoritmilor

### Comparatie obiectiva

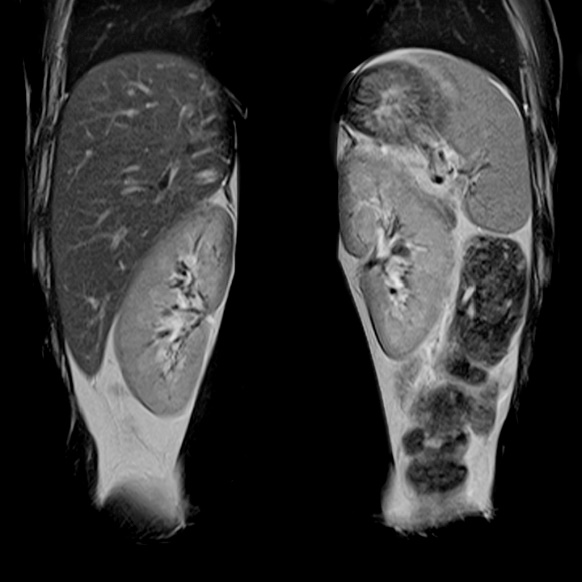
Comparatia obiectiva a calitatii algoritmilor se efectuează cu ajutorul rezultatelor procesului de masurarea calitatii, mai precis datele salvate in fisierul Excel. Pentru fiecare set de imagini de intrare vom afisa valorile cele mai bune din punctul de vedere al MSE si PSNR-ului.

#### Imaginea perfecta



Figură 6.1 imaginea perfecta

#### Setul de imagini 1

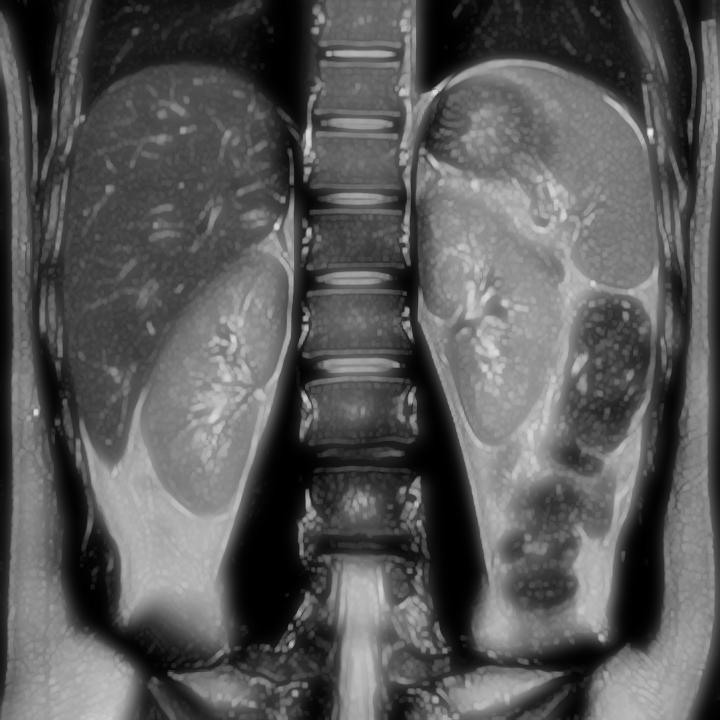
Figură 6.2 Stanga: mri\_soft\_2.jpg, dreapta: mri\_hard\_2.jpg

Rezultatele cele mai bune obtinute din results.xls:

|  |  |  |
| --- | --- | --- |
| **Algoritm\_numeimagini\_postprocesare** | **MSE** | **PSNR** |
| Maximum mri\_soft\_2 + mri\_hard\_2\_eroded | 609.8186 | 20.2788 |
| Laplacian\_3\_3.0 mri\_soft\_2 + mri\_hard\_2\_eroded | 725.6997 | 19.52323 |
| Haar\_2 mri\_soft\_2 + mri\_hard\_2\_smoothed | 787.247 | 19.16969 |



Figură 6.3 Maximum mri\_soft\_2 + mri\_hard\_2\_eroded

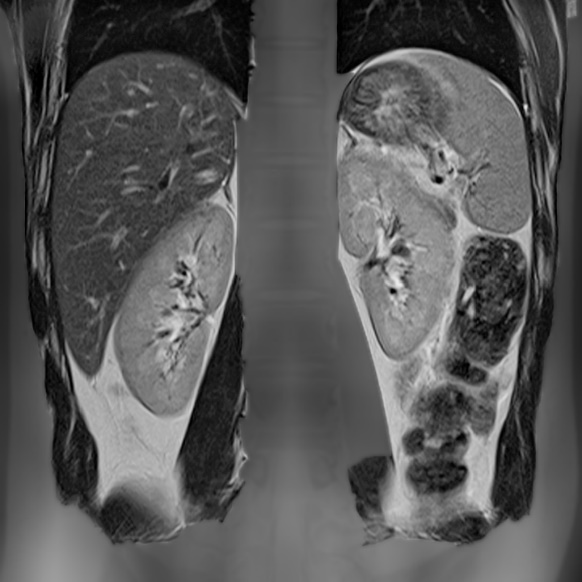
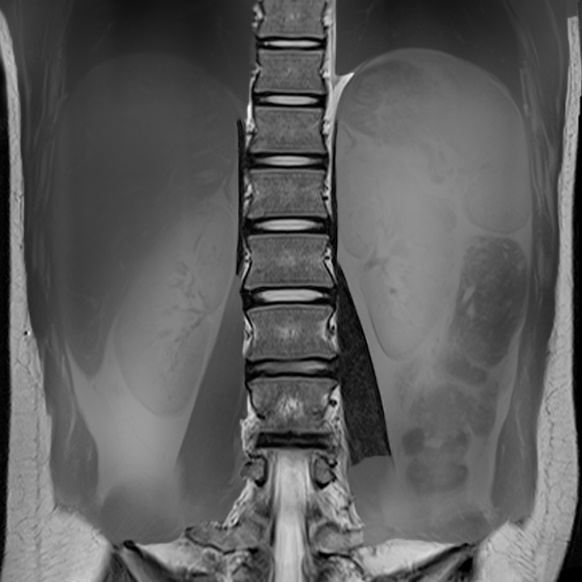


Figură 6.4 Laplacian\_3\_3.0 mri\_soft\_2 + mri\_hard\_2\_eroded



Figură 6.5 Haar\_2 mri\_soft\_2 + mri\_hard\_2\_smoothed

#### Setul de imagini 2

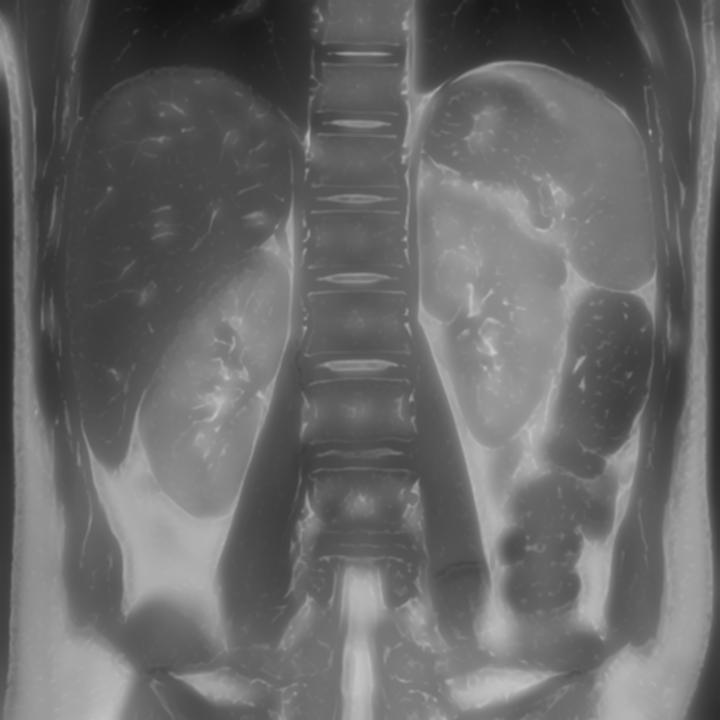
 

Figură 6.6 Stanga: mri\_soft.jpg, dreapta: mri\_hard.jpg

|  |  |  |
| --- | --- | --- |
| **Algoritm\_numeimagini\_postprocesare** | **MSE** | **PSNR** |
| Average mri\_soft + mri\_hard | 421.3322 | 21.88456 |
| Haar\_2 mri\_soft + mri\_hard\_dilated\_smoothed | 531.9417 | 20.87216 |
| Maximum mri\_soft + mri\_hard | 640.0845 | 20.06843 |



Figură 6.7 Average mri\_soft + mri\_hard

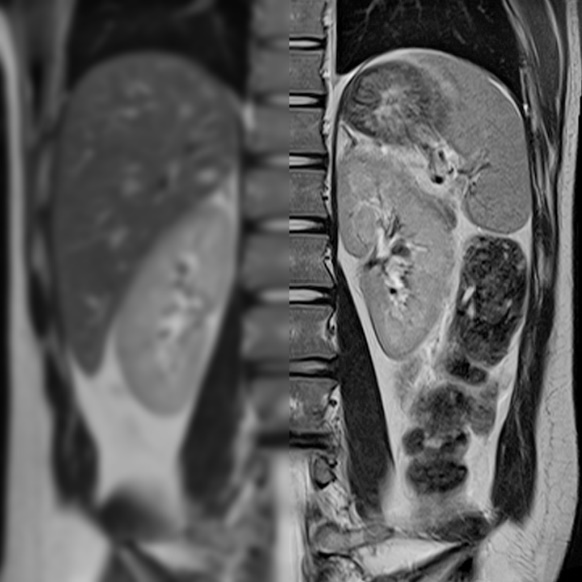
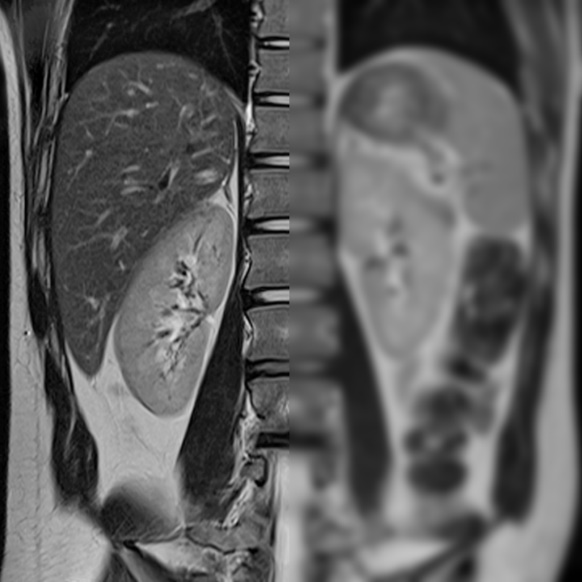


Figură 6.8 Haar\_2 mri\_soft + mri\_hard\_dilated\_smoothed



Figură 6.9 Maximum mri\_soft + mri\_hard

#### Setul de imagini 3

Figură 6.10 Stanga: mri\_left\_blurred.jpg, dreapta: mri\_right\_blurred.jpg

|  |  |  |
| --- | --- | --- |
| **Algoritm\_numeimagini\_postprocesare** | **MSE** | **PSNR** |
| Average mri\_left\_blurred + mri\_right\_blurred | 123.3515 | 27.21936 |
| Haar\_2 mri\_left\_blurred + mri\_right\_blurred\_dilated\_eroded | 190.5589 | 25.33051 |
| Maximum mri\_left\_blurred + mri\_right\_blurred | 237.214 | 24.3794 |



Figură 6.11 Average mri\_left\_blurred + mri\_right\_blurred



Figură 6.12 Haar\_2 mri\_left\_blurred + mri\_right\_blurred\_dilated\_eroded



Figură 6.13 Maximum mri\_left\_blurred + mri\_right\_blurred

#### Media rezultatelor

Generalizand rezultatele dupa tipul de algoritmi, putem calcula media PSNR-ului:

1. Haar: 21.78
2. Maxim aritmetic: 21.56
3. Medie aritmetica: 20.68
4. Laplacian: 19.79
5. Minim aritmetic: 16.61

Observatie importanta despre aceste rezultate: in cazul in care imaginil de intrare nu sunt complementare, cum este in cazul primului set de imagini, metodele aritmetice produc rezultate foarte proaste.

### Comparație subiectiva

### Rezultate

## Teste de unitate

# Manual de Instalare si Utilizare

## Instalare

Sistemul pe care se doreste rularea aplicatiei trebuie sa indeplineasca urmatoarele cerinte:

### Cerințe hardware

Aplicatia fiind relativ mica, procesele ei nu necesita putere foarte mare de calcul. O configuratie minima pe care ruleaza un sistem de operare modern este capabil sa ruleze si aplicatia fara probleme, singura chestie afectata ar putea fi durata proceselor de fuziune sau masurarea calitatii. Totusi, orientativ, specificam urmatoarele cerinte minime:

* Procesor Intel® Pentium™ de 2 GHz sau mai bun
* Memorie RAM 2GB sau mai mult
* Saptiu de stocare minim 10MB pentru aplicatie, si 20 de MB pentru rezultate

### Cerințe software

Independenta de platforma a proiectului ofera posibilitatea folosirii oricarui sistem de operare preferat de catre utilizator.

Aplicatia fiind scrisa in limbajul de programare Java™, avem nevoie de un mediu Java Runtime Environment pentru a o rula. Acest JRE™ se poate instala de pe pagina web [[[27]](#endnote-27)], selectand sistemul de operare si arhitectura aferenta. Dupa instalare se poate deschide fisierul jar al aplicatiei.

Daca se doreste dezvoltarea aplicatiei, construirea ei din codul sursa, trebuie instalat un Java Development Kit, sau JDK™. Acesta se poate instala de pe [[[28]](#endnote-28)], selectand, la fel, optiunile corecte la descarcare. Codul sursa al proiectului se poate clona local de pe GitHub, de pe adresa <https://github.com/a-henning/DicomFusion>.

De observat folosirea Maven-ului pentru descarcarea dependintelor proiectului si automatizarea procesului de construire. Aceasta unealta poate fi deja integrata in editor-ul de cod sursa, sau poate fi descarcata de pe pagina proiectului [14].

## Utilizare

# Concluzii

# Bibliografie

1. Imagistica Medicala, Wikipedia, <http://ro.wikipedia.org/wiki/Imagistic%C4%83_medical%C4%83> [↑](#endnote-ref-1)
2. A. A. Feiler, A.-M. Ungureanu, „Manual de ragiologie si imagistica medicala” [↑](#endnote-ref-2)
3. DICOM, Wikipedia, <http://en.wikipedia.org/wiki/DICOM> [↑](#endnote-ref-3)
4. Deepak Kumar Sahu, M.P. Parsai, “Different Image Fusion Techniques – A Critical Review” [↑](#endnote-ref-4)
5. Image Fusion, Wikipeda, <http://en.wikipedia.org/wiki/Image_fusion> [↑](#endnote-ref-5)
6. Mirada Medical XD3, <http://www.mirada-medical.com/products/xd3/> [↑](#endnote-ref-6)
7. Velocity Medical, <http://www.velocitymedical.com/solutions/> [↑](#endnote-ref-7)
8. Keosys Imagys-Cloud Services, <http://www.keosys.com/eng/visualize/workstation_nm.php> [↑](#endnote-ref-8)
9. ImageJ, <http://imagej.nih.gov/ij/features.html> [↑](#endnote-ref-9)
10. JExcel API, <http://jexcelapi.sourceforge.net/> [↑](#endnote-ref-10)
11. Lesson: Learning Swing with the NetBeans IDE, <http://docs.oracle.com/javase/tutorial/uiswing/learn/> [↑](#endnote-ref-11)
12. JUnit, <http://junit.org/> [↑](#endnote-ref-12)
13. JUnit, Wikipedia, <http://en.wikipedia.org/wiki/JUnit> [↑](#endnote-ref-13)
14. Apache Maven, <http://maven.apache.org/> [↑](#endnote-ref-14)
15. DICOM Specification Overview: Basic DICOM File Structure, <http://www.leadtools.com/sdk/medical/dicom-spec1.htm> [↑](#endnote-ref-15)
16. C. Melenti, “Formate de imagine”, curs Tehnologii Multimedia [↑](#endnote-ref-16)
17. MATLAB Image Compression using Haar Wavelet Transform, <http://www.eeweb.com/electronics-forum/matlab-image-compression-using-haar-wavelet-transform> [↑](#endnote-ref-17)
18. P. Jorgensen, „Image Decomposition using Haar Wavelet”, <http://homepage.math.uiowa.edu/~jorgen/Haar.html> [↑](#endnote-ref-18)
19. S. Ludwig, „Implementation of a spatio-temporal

    Laplacian image pyramid on the GPU”, <http://www.gazecom.eu/FILES/ludw08.pdf> [↑](#endnote-ref-19)
20. R. Danescu, “Operatii morfologice”, curs Procesarea Imagnilior [↑](#endnote-ref-20)
21. R. Fisher, S. Perkins, A. Walker and E. Wolfart, „Dilation”,

    <http://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm> [↑](#endnote-ref-21)
22. Dilation (morphology), Wikipedia, <http://en.wikipedia.org/wiki/Dilation_(morphology)> [↑](#endnote-ref-22)
23. R. Fisher, S. Perkins, A. Walker and E. Wolfart, „Erosion”,

    <http://homepages.inf.ed.ac.uk/rbf/HIPR2/erode.htm> [↑](#endnote-ref-23)
24. Erosion (morphology), Wikipedia, <http://en.wikipedia.org/wiki/Erosion_(morphology)> [↑](#endnote-ref-24)
25. Smoothing, Wikipedia, <http://en.wikipedia.org/wiki/Smoothing> [↑](#endnote-ref-25)
26. ImagePlus, ImageJ API, <http://imagej.nih.gov/ij/developer/api/ij/ImagePlus.html> [↑](#endnote-ref-26)
27. Oracle, „Java SE Runtime Environment 7 Downloads”, <http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html> [↑](#endnote-ref-27)
28. Oracle, “Java SE Downloads”, <http://www.oracle.com/technetwork/java/javase/downloads/index.html?ssSourceSiteId=otnjp> [↑](#endnote-ref-28)