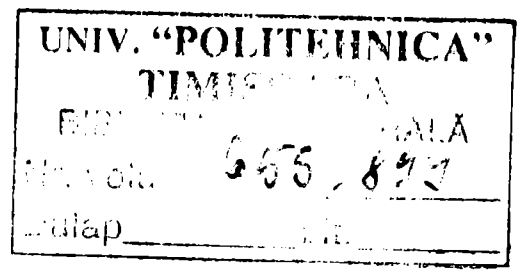
STUDIUL MĂRIRII CAPACITĂŢILOR OPERAŢIONALE ALE MAŞINILOR DE PROTOTIPARE RAPIDĂ PRIN PRELEVARE DE MATERIAL

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul INGINERIE INDUSTRIALĂ
de către

Ing. Marius-Iulian Tamas

Conducător ştiinţific: Prof.Dr.Ing. Tudor Iclănzan
Referenţi ştiinţifici: Prof.Dr.Ing Petre Berce
Prof.Dr.Ing. Nicolae Bâlc
Prof.Dr.Ing. George Savii

Ziua susţinerii tezei: 21.03.2008



Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor

Universitatea „Politehnica" din Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Dascălilor mei

Cuvânt înainte

Teza de doctorat a fost elaborată în cadrul Catedrei de Tehnologia Construcţiilor de Maşini a Universităţii „Politehnica" din Timişoara.

Tamas, Marius-Iulian

Studiul măririi capacităţilor de producţie ale maşinilor de prototipare rapidă prin prelevare de material

Teze de doctorat ale UPT, Seria 8, Nr. 7, Editura Politehnica, 2008, 210 pagini, 259 figuri, 15 tabele.

ISSN: 1842-8967

ISBN: 978-973-625-611-0

Cuvinte cheie: Prototipare rapidă, Computer Vision, Prelucrarea Imaginilor Digitale, G-Code, Blender, Vizualizare, Simulare, Comandă Numerică, EMC2, Python, Java, Reverse Engineering, Open Source, GRID

Rezumat: Scopul principal al acestei lucrări este de a acumula cât mai multe informaţii referitoare la aplicarea şi utilizarea nu numai a metodelor ci şi algoritmilor şi modelelor matematice existente în alte domenii în vederea prototipării rapide prin prelevare de material. Un alt scop este cel al punerii la dispoziţie în domeniul prototipării rapide de noi utilitare şi sisteme de lucru, în vederea obţinerii unei accesibilităţi ridicate, atât pentru cadrele specializate şi educaţionale cât şi pentru domeniul firmelor mici, mijlocii sau chiar pentru domeniul privat. Considerând prototiparea rapidă ca având loc de la ideea iniţială (imagini digitale, schiţe) şi nu doar redusă la o simplă fabricaţie rapidă a prototipurilor, lucrarea încearcă să redea un fir roşu al posibilităţilor măririi capacităţilor productive ale maşinilor de prototipare rapidă prin prelevare de material, ca urmare a generării automate de G-Code pornind de la imagini digitale. De asemenea, reducerea costurilor de producţie (material şi timp) prin vizualizarea şi simularea rezultatelor (prototipare virtuală), precum şi prin exclusiva utilizare a utilitarelor de tip Open Source. Neîncercând să diminueze importanţa utilitarelor şi sistemelor profesionale în vederea prototipării rapide, lucrarea demonstrază validitatea, realizabilitatea şi fiabilitatea dezideratelor propuse prin implementări şi prelucrări reale a rezultatelor obţinute, în special în vederea utilizării lor pentru domenii care, la ora actuală, nu au acces larg la utilitare profesionale - în principal datorită investiţiilor iniţiale ridicate (artă, manufactură şi artizanat, hobby şi mai ales în scopuri educaţionale). În vederea prelucrării distribuite a imaginilor de dimensiuni mari şi/sau a detectării cercurilor şi imaginilor de referinţă de dimensiuni şi orientări variabile, utilizarea algoritmilor adoptaţi şi adaptaţi în sisteme distribuite a fost de asemenea tratată.

Cuprins

1 Motivaţie şi scop

Din moment ce elemente ca Analiza Imaginilor digitale, Computer Vision şi Pattern Recognition sunt tot mai larg răspândite în domenii ca robotica, modelare, vizualizare şi simulare arhitectonică, apare intuitiv problema utilizării algoritmilor existenţi (bineânţeles cu modificările şi adaptările necesare) în domeniul optimizării prototipării rapide.

La ora actuala, elementele hard necesare (camera video, laser, computer) sunt tot mai accesibile, iar algoritmii necesari sunt tot mai evoluaţi şi performanţi. Ca urmare a dezvotării explozive a domeniului Open-Source, tot mai multe institute de cercetare, universităţi şi persoane private adoptă acest mod de lucru datorită numeroaselor avantaje oferite.

Scopul principal al acestei lucrări este de a acumula cât mai multe informaţii referitoare la aplicarea şi utilizarea nu numai a metodelor ci şi algoritmilor şi modelelor matematice existente. Un alt scop este cel al punerii la dispoziţie în domeniul prototipării rapide de noi utilitare şi sisteme de lucru în vederea obţinerii unei accesibilităţi ridicate, atât pentru cadrele specializate şi educaţionale cât şi pentru domeniul firmelor mici, mijlocii sau chiar pentru domeniul privat.

Unele din metodele prezentate în cadrul lucrării sunt uneori utilizate şi în programe comerciale, în special în cazuri în care alte metode sunt mult mai costisitoare sau chiar imposibile (suprafete geografice în cazul modelarii unei hidrocentrale, analiza modului de distributie a curenţilor de aer în cazul modificării architectonice a unei străzi sau a unei camere, modelarea unei baze extraterestriale pe Lună sau Marte pornind de la imagini prin sateliţi, modelarea şi vizualizarea fluidelor prin ventile în vederea îmbunătăţirii curgerii, generarea de proteze medicale pe baza imaginilor extrase din radiografii şi multe altele). Pe de altă parte însă, aceste metode se pretează chiar şi în cazul reingineriei unor piese de uz privat - de exemplu ieşite din procesul de producţie, sau a căror procurare ar fi mult prea costisitoare.

Un alt capitol deosebit de important la ora actuală este cel dedicat vizualizării şi simulării. Modelele matematice existente permit o tot mai buna aproximare a realităţii, reducând tot mai mult costurile legate de procesul de fabricatie prin eliminarea atât a erorilor de proiectare cât şi a celor legate de testare (teste de functionare şi crash).

Utilizarea programelor comerciale existente este incontestabil extrem de utilă, dar ţinând cont de modul exploziv de dezvoltare şi evoluţie atât a utilajelor de strunjire şi frezare cu comanda numerică, cât şi a tehnicii de calcul necesare, dependenţa unui domeniu atât de vast de un numar redus de programe şi utilitare comerciale nu este totdeauna avantajos. Este adevărat că prin specializarea pe o anumita maşină şi un anumit program, productivitatea este sau poate fi maximizată, dar pe de altă parte duce la un grad de inflexibilitate care, nu numai în cazul firmelor mici şi mijlocii ci şi în cazul concernelor, poate deveni fatală.

Pe de altă parte, dacă ţinem cont de faptul că nu numai tehnica de calcul a evoluat într-un ritm alert în ultimii ani ci, datorită reţelelor de calculatoare şi în special a Internetului, modul şi ritmul de prelucrarea a informaţiilor au fost de-a dreptul revoluţionate, prototiparea rapidă a început să fie o şansă pentru toţi cei care vor să-şi testeze ideile sau să-şi îmbunatăţească cunoştiinţele în domenii tehnice. Multe dintre cele prezentate în continuare sunt încă într-un stadiu incipient de cercetare, ceea ce însa nu le reduce de loc gradul de importanţă, căci ceea ce acum 20 de ani utopie sau imposibil părea, a devenit azi realitate prin realizarea unor siteme de Realitate Virtuala (VR), CAVE sau chiar Realitate Virtuala Extinsă (AVR). Chiar dacă la ora actuală sistemele de calcul necesare pentru acest tip de vizualizari şi simulari depăşesc de cele mai multe ori posibilităţile reale ale persoanelor fizice sau firmelor mici - să nu uităm că nu de mult, puterea unui centru de calcul era de cateva ori mai mică decât a celui mai puţin performant PC cumpărabil la ora actuală pe piaţă.

În general însă, **capacitatea de prelucrare a maşinilor de prototipare rapidă prin prelevare de material accesibile este extrem de redusă**, mai ales datorită faptului că fiind la un preţ accesibil, avantajele maşinilor profesionale nu vor putea fi în curând atinse – corecţie şi schimbare automata de scule, calibrare şi poziţionare automată, controlul şi stabilizarea turaţiei, conectarea automată la sisteme CAM/ DCAM. Cu toate acestea, capacitatea lor poate fi optimal utilizată prin reducerea timpului total necesar procesului de prototipare rapidă în fiecare fază a acestuia, începând de la scanare (în cazul în care prototiparea rapidă

face parte dintr-un proces de reverse engineering) şi până la simularea procesului şi obţinerea unei piese virtuale.

Asigurând o flexibilitate ridicată prin elemente de postprocesare care să permită adaptarea la cazuri particulare de maşini de frezat sau strunjit cu comandă numerică (G-Code), generalitatea sistemului prezentat, utilizat şi îmbunătăţit în cadrul lucrării de faţă poate fi păstrată, iar utilizarea sa în domeniul educaţional, privat şi în firmele mici şi mijlocii va permite o creştere remarcabilă a productivităţii la o investiţie minimă. Pe de altă parte, deşi marea majoritate a algoritmilor şi modelelor matematice prezentate nu sunt specifice numai prototiparii rapide prin prelevare de material, prezentarea acestora este necesară în vederea scanării, modelării sau vizualizării procesului.

Fiind un sistem bazat pe conceptul de Open Source, este evident că ritmul şi calitatea sa depinde inevitabil atât de numarul programatorilor participanţi cât şi de timpul investit de aceştia în procesul dezvoltării modulelor. Avantajul deosebit şi totodată impedimentul principal este dat însă de faptul că deciziile - atât în vederea direcţiei de dezvoltare cât şi a eliberării de noi versiuni - nu sunt dictate decât de participanţii la dezvoltarea programelor. Din acest motiv, este foarte posibil, ca dezvoltarea acestor sisteme să devină un sistem perpetuu, adăugarea de noi module, finisarea şi adaptarea celor existente fiind un date de modul şi necesitatea dictată de piaţă. De amintit trebuie însă în orice caz şi riscul ce apare din aceleaşi motive – un proiect, a cărui repere nu sunt dictate decât de bunăvoinţa şi motivaţia programatorilor participanţi, poate duce la oprirea bruscă a dezvoltării sistemului, fie datorită complexităţilor apărute, fie datorită lipsei de timp necesar sau chiar a motivaţiei.

Unul dintre ţelurile principale ale sistemului prezentat este independenţa de sistemul de operare, prin utilizarea exclusivă de utilitare de acest tip (utilitare, interpretoare, front-ends, compilatoare). Astfel, dezvoltarea de programe are şi va avea loc în Java şi Python, mijlocul de legătură dintre utilitare fiind bash/ksh (specifice Unix/Linux, dar şi sub Windows fiind instalabile prin intermediul pachetului cygwin), interpretoare existente sub marea majoritate a sistemelor de operare existente, iar simularea şi vizualizarea prin intermediul programului Blender, care de asemenea respectă aceste condiţii. Cu toate acestea, dat fiind faptul că atât strungul cu comandă numerică din posesie, cât şi freza utilizată pentru validarea rezultatelor sunt controlate utilizând sistemul de operare Linux - datorită atât facilităţilor de operare în timp real ale acestuia cât şi faptului că acest sistem, fiind tot de tip Open Source, se integrează perfect în cadrul conceptului general al lucrării. Utilitarele prezentate în continuare au fost testate în mod exclusiv sub acest sistem de operare – chiar dacă ele există în aceeaşi variantă şi sub Windows. Motivul principal a fost preţul suplimentar al versiunii Windows a părţii de control a comenzii numerice.

Lucrarea de faţă va încerca aşadar să dovedească faptul că utilizarea tehnicilor, metodelor, algoritmilor şi modelelor matematice specifice prelucrării imaginilor digitale pot fi adaptate şi utilizate cu succes în prototiparea rapidă, obţinându-se în acest fel nu numai o creştere a capacităţilor operaţionale a maşinilor ci – datorită reducerii semificative a preţului prin utilizarea de OpenSource - şi o largă deschidere a metodelor şi tehnicilor utilizate pentru utilizarea lor în domenii mai puţin răspândite (hobby, artă). De asemenea, simularea şi vizualizarea rezultatelor obţinute, alături de utilizarea tehnicilor de distribuire a puterii de calcul, sunt premizele obţinerii unor rezultate tot mai optimale în domeniul prototipării rapide.

Evident că fiind primii paşi în această direcţie, implementarea este cât se poate de didactică, optimizarea algoritmilor utilizaţi şi perfecţionarea lor nefăcând parte din scopul lucrării. Într-o lume care evoluează tehnic şi tehnologic într-un ritm exploziv, inerţia este începutul sfârşitului - Mark Twain „If the only tool you have is a hammer, every problem looks like a nail".

2 Introducere

Datorită dezvoltării şi schimbării rapide a pieţelor de desfacere, necesitatea aducerii pe piaţă a produselor noi într-un timp cât mai rapid ('time to market'), tehnologiile de design şi producţie au evoluat în ultimii ani într-un ritm extrem de rapid. Dintre elementele de bază ce permit reducerea acestui timp, pot cu siguranţă fi menţionate ingineria simultană (Concurent Engineering), proiectarea asistată de calculator (Computer Aided Design – CAD), ingineria asistată de calucator (Computer Aided Engineering – CAE), fabricaţia asistată de calculator (Computer Aided Manufacturing – CAM), prototiparea rapidă (Rapid Prototyping – RP) şi prototiparea virtuală (Virtual Prototyping – VP). Aceste elemente permit concomitent şi introducerea pe piaţă a unor produse tot mai apropiate de cerinţele consumatorilor (tailor product to customer), reducerea costurilor de producţie, paralel cu îmbunătăţirea permanentă a calităţii produselor. Ţinând cont de faptul că pretenţiile pieţei sunt într-o creştere continuă, în primul rând de preţ şi calitate, şi într-o perioada în care sloganul ce reprezintă cel mai concret realitatea este 'change or die', întreprinderile timpurilor noastre sunt practic obligate să se adapteze metodelor tehnologice noi, din care o parte vor fi prezentate în lucrarea de faţă.

Dezvoltarea produsului în cadrul inginerie simultane este schematic descrisă în Fig. 1:

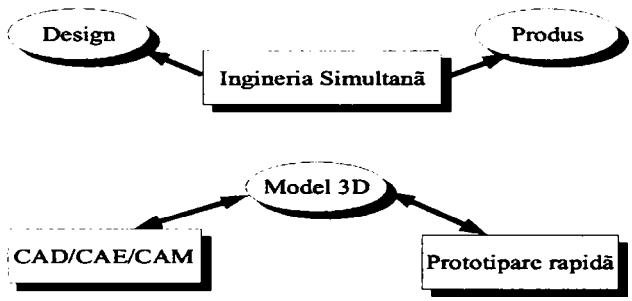


Fig. 1 Ingineria Simultană (Concurent Engineering)

Această grupă de tehnologii, apărută în anii '90, s-a extins cu succes asupra tuturor domeniilor de activitate umană. Tehnologii şi mijloace de prototipare rapidă există atât în domeniul ingineriei mecanice cât şi în chimie, electrotehnică şi electronică, precum şi în informatică, unde pe de altă parte, dezvoltarea de programe fără metode de prototipare rapidă nu mai este de conceput. În principal, prototiparea rapidă (Rapid Prototyping - RP) este rezultatul dezvoltării explozive a sistemelor de calcul şi a tehnologiilor ce permit desfăşurarea interdisciplinară a activităţilor de producţie. Prin utilizarea metodelor numerice de analiză a elementelor finite se realizează concomitent o reducere substanţială de timp şi material, dat fiind faptul că numeroase teste din faza de testare a prototipului sunt simulate în cadrul modelării.

2.1 Cadrul general al prototipării rapide

Toate tehnologiile urmează în mare aceleaşi etape în realizarea prototipării rapide [1]:

1. Modelarea CAD.
2. Transferul modelului la unitatile de secţionare, prin care suprafeţele sunt aproximate în vederea prelucrării. Metoda de aproximare cel mai mult aplicată este cea de triangularizare, prin care suprafaţa obiectului este aproximată prin elemente triunghiulare plane.
3. Modelul este pregătit pentru secţionare şi construcţie.
4. Construcţia
   În cadrul acestui pas, apar probleme specifice fiecărui proces, cum ar fi:
   - alegerea materialului ce va fi folosit
   - fixarea şi prinderea materialului în vederea construcţiei

- determinarea straturilor de prelucrare
- ...

5. Curățirea și finisarea în vederea îmbunătățirii preciziei dimensionale

Spre deosebire de metodele de fabricație standard, marea majoritate a metodelor de prototipare rapidă se bazează pe adăugarea de material (ARP) și nu pe detașarea materialului (SRP) în vererea obținerii piesei finite. Din această cauză, în primii ani de existență ai prototipării rapide, elaborarea unui prototip sau model era realizabilă în decursul unei săptămâni, pe când durata aceeași prelucrări prin metode clasice era de două până la patru ori mai mare. În utlimii ani însă, prin dezvoltarea tehnologiilor și mașinilor de prelucrare prin detașare de material, caracterizarea prototipării rapide ca o metodă aditivă de prelucrare este din ce în ce mai mult contestată.

Ciclul de definire a produsului, pe de altă parte poate fi definită ca fiind caracterizată de două faze, o fază de design propriu-zisă, și una de optimizare a produsului [2]. Schematic aceste două faze sunt descrise în Fig. 2.

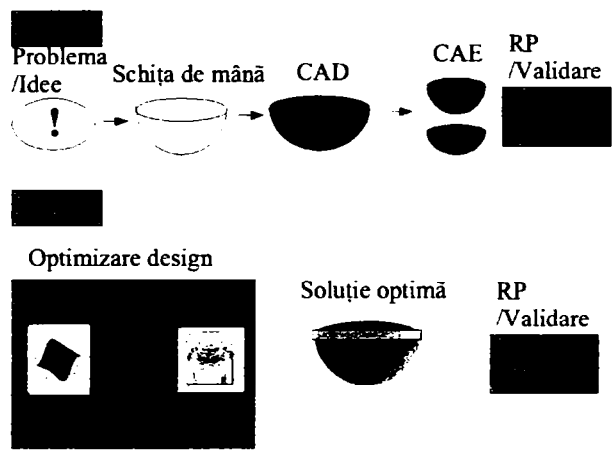


Fig. 2 Fazele ciclului de definire a produsului

Un alt domeniu, la fel de important care trebuie menționat, este cel de Reverse Engineering, strîns legat de prototiparea rapidă, dat fiind faptul că prin acest pro- -s -- t---- d- l- p--d-s- -xist---, ---l , la fo m lo mod l t în l ul t . Pentru majoritatea produselor și pieselor existente, în cazul în care nu au fost rezultatele unui process tipic de prototipare rapidă, nu există modelul CAD, iar pen ru forme complexe, o mo 'e'are manuală este costisitoare. Astfel, reducerea timpului de modelare, prin utilizarea de exemplu a utilajelor cu palpatoare sau scannere 3D, este semificativă.

2.2 *Avantaje*

Datorită posibilității de vizualizare a modelelor CAD și a utilizării programelor de interfață între programul CAD și mașinile-unelte NC sau CNC, numeroase avantaje pot fi puse în evidență:

1. pregătirea în avans a lansării produsului – începând de la marketing și până la vânzare – cu mult timp înainte de realizarea primului produs
2. posibilitatea de vizualizare a produsului, în cazuri foarte complexe prin metode de Realitate Virtuala (Virtual Reality - VR), ajută la elaborarea elementelor de design, luarea deciziei de promovare a produsului și uneori la realizarea machetelor în vederea susținerii propunerii de produs
3. reducerea costurilor de producție – cu toate că raportul preț/performanță era de așteptat să se îmbunătățească, reducerea unităților de timp pentru aceeași piesă în 1992 de la 15 unități de timp la 2 unități de timp în 2000 a depășit surprinzător și totodată spectaculos toate așteptările
4. îmbunătățirea evaluării produsului, prin eliminarea principalului impediment în calea unei evaluări rapide – realizată pe baza desenelor.
5. reducerea riscurilor de apariție a necesităților de modificare a produsului în faza de producție, modificări care întotdeauna sunt legate de creșterea costurilor de producție dat fiind faptul că orice modificare atrage după ea modificări de SDV-uri
6. modelele pot fi cu ușurință testate în diverse moduri (în funcție de sistemele în dotare – CAD / CAE /

CAM):
- teste funcționale
- teste de simulare
- teste de control
- teste de fabricație
- teste de fixare și asamblare
- teste de ambalare

7. posibilitatea simulării comportării produsului în diverse variante constructive și/sau materiale

2.3 Clasificarea tehnologiilor de fabricație rapidă a prototipurilor

Una dintre posibilitățile de clasificare este după procedeele de prelucrare, descrise în Fig. 3.

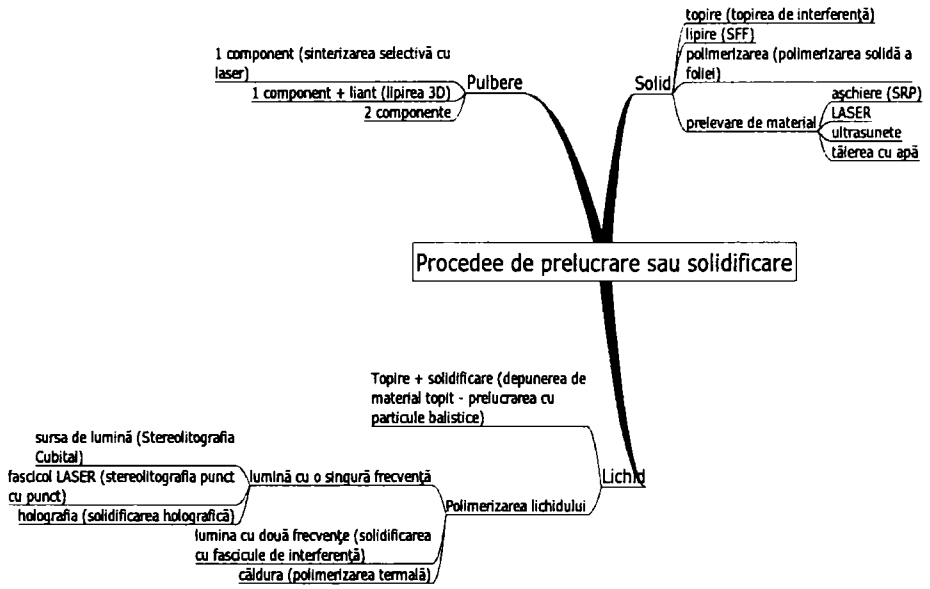


Fig. 3 Procedee de prelucrare

O altă posibilitate de clasificare este după metoda de realizare a formei, schematic descrisă în Fig. 4.

Din totalitatea domeniilor în care prototiparea rapidă a câștigat în importantă, în cadrul lucrării de față ne vom referi numai la tehnologiile legate de prototiparea rapidă prin prelevare de material.

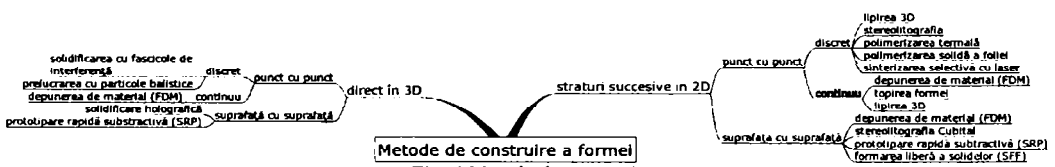


Fig. 4 Metode de construire

Chiar dacă lipirea (fabricarea de piese stratificate) nu pare a avea legătură cu prelevarea de material, trebuie remarcat faptul că straturile nu pot fi adăugate fără o prealabilă prelevare a zonelor ce nu aparțin stratului actual. Ca urmare a acestei constatări, precum și a faptului că fabricarea de piese stratificate este una

dintre cele mai importante modalităţi (iar de foarte multe ori singura) de realizare a prototipurilor şi pieselor de complexitate mare, ne vom ocupa în continuare doar de SRP şi SFF.

Dat fiind faptul că pentru compararea sistemelor de prototipare rapidă prin prelevare de material o importanţă deosebită prezintă nu numai caractersticile tehnice ale maşinilor, ci şi programele existente şi accesibile, ne vom ocupa pe scurt şi de:

- programele de CAD, CAE, CAM livrate sau cu care sistemele pot fi interfaţate
- posibilităţile de interfaţare cu alte programe (FEM, VR, VP) în vederea analizei şi simulării deformărilor şi tensiunilor ce apar în materiale în urma acţiunilor exterioare, vizualizarea în vederea discuţiilor de design, producţie, marketing, alegerea materialelor, etc.

2.4 Prototiparea rapidă ca parte integrantă a procesului de RE

Dată fiind dezvoltarea şi schimbarea rapidă a pieţelor de desfacere, necesitatea aducerii pe piaţă a produselor noi într-un timp cât mai rapid ('time to market'), tehnologiile de design şi producţie au evoluat în ultimii ani într-un ritm extrem de rapid. Una dintre cerinţele şi totodată necesitatea de bază a asigurării perenităţii unei întreprinderi este deci adaptarea cât mai rapidă la necesităţile consumatorilor, determinată în principal de flexibilitatea ei, unul dintre sloganurile cele mai actuale fiind 'change or die'.

Datorită noilor tehnologii, materiale şi evoluţiei explozive a tehnicii de calcul, în ultimul timp întreprinderile sunt tot mai mult obligate să utilizeze metode şi tehnici ale ingineriei simultane (şi/sau integrate) în vederea reducerii timpului de aducere a unui nou produs pe piaţă.

În mod normal, ingineria este definită ca fiind procesul de descoperire a legilor şi principiilor de funcţionare a a unui sistem şi materializarea/aplicarea lor în vederea creării unui artefact care satisface o anumită necesitate. S-a constatat însă că în majoritatea cazurilor produsele sunt doar îmbunătăţite sau remodelate după un produs existent, ceea ce a dus la definirea noţiunii de Reverse Engineering (RE).

În [3] articolul 'The Law and Economics of Reverse Engineering' este prezentată următoarea definiţie – 'Reverse Engineering ... broadly speaking, is the process of extracting know-how or knowledge from a human made artifact.'

O altă definiţie, mai completă şi mai corectă, poate fi găsită în 'Reverse Engineering and Design Recovery: A Taxonomy' [4] - Reverse Engineering este procesul de analizare a unui sistem subiect în vederea identificării componentelor sale precum şi a interacţiunilor dintre acestea, precum şi crearea unei reprezentări a sistemului într-o altă formă sau la un nivel ridicat de abstractizare.

Aşadar, Reverse Engineering cuprinde în general extragerea anumitor elemente de design şi construirea sau sintetizarea lor abstractă, făcându-le astfel independente de modul de producere sau implementare.

Cu toate că Reverse Engineering este de cele mai multe ori bazată pe existenţa unui sistem subiect funcţional, această condiţie nu este obligatorie şi nici necesară, tehnicile de Reverse Engineering putând fi aplicate la orice nivel de abstractizare şi în orice etapă a unui ciclu de viaţă. Există astfel tehnici de Reverse Engineering, sau cel puţin există posibilitatea de aplicare a lor, în toate domeniile de activitate umane.

Trebuie de asemenea reţinut faptul, că Reverse Engineering este doar un proces de examinare şi nu de schimbare sau duplicare a sistemului iniţial. Sistemul analizat nu va fi nici influienţat, nici schimbat şi nici replicat prin aplicarea RE, acţiunile amintite făcând parte din tehnicile de Reengineering şi care trebuie clar diferenţiate de RE.

Ţinând cont de cele amintite mai sus, putem generaliza faptul că de fapt toată istoria umană este bazată pe RE, omul fiind singura fiinţă de pe planetă care posedă o gândire abstractă, suficient de evoluată care să-i permită adaptarea mediului înconjurător după nevoile sale – iar uneori, din nefericire după bunul său plac, fără a încerca să analizeze şi repercursiunile cauzate de acţiunile sale. Iniţial au existat doar obiectele naturale, care cu timpul, în urma unui proces cognitiv voluntar sau involuntar de RE au fost 'perfecţionate' în vederea atingerii anumitor scopuri.

2.4.1 Istoric

Cu toate că Reverse Engineering ca și concept este nou, principiul și tehnicile au apărut și s-au dezvoltat odată cu specia umană. Dacă am încerca să împărțim istoria umană în etape de utilizare a tehnicilor de RE, am putea defini următoarele etape:

2.4.1.1 I - Etapa/nivelul abstractizărilor empirice (asemănarea formei)

Apariția primelor unelte și arme a atras cu siguranță după sine dorința de perfecționare, ceea ce implică o analiză mai mult sau mai puțin complexă a sistemului utilizat - chiar dacă cele mai multe îmbunătățiri au apărut în mod accidental. Astfel, de la simpla observație că piatra de râu spartă taie mai bine decât cea veche, a condus la una din primele abstractizări – omul primitiv a început să dea mai mare importanță pietrelor de o anumită formă și care deci puteau fi mai ușor ascuțite. În această etapă, majoritatea descoperirilor aveau loc accidental, analiza cauzală și deterministică având totuși un rol deosebit de important cu toate că explicațiile fenomenelor erau bazate pe credințe oculte sau religioase. Spiritul inovativ uman însă a continuat să se dezvolte, trecând cu succes prin fazele accidentale ale descoperirii focului și facilităților acestuia în viața de zi cu zi, a descoperirii roții, arcului cu săgeți. Cu timpul procesele de Reverse Engineering și Reengineering au fost continuu aplicate în mod secvențial și iterativ în vederea îmbunătățirii sistemelor (produselor) obținute. O adevărată revoluție a apărut în momentul descoperirii metalelor, când partea de RE a devenit tehnica principală utilizată. Negativele obiectelor existente au fost obținute prin presarea lor în lut, permițând turnarea metalelor în forma astfel obținută. Astfel a fost posibilă, pentru prima dată în istoria umanității, abstractizarea prin modelare fizică a unui obiect și totodată modificarea caracteristicilor sistemului inițial, dat fiind faptul că singura asemănare între obiectul inițial și cel final era forma exterioară.

2.4.1.2 II - Etapa/nivelul abordărilor științifice (asemănarea fizică)

În această etapă apar primele încercări de abordări științifice în special în domeniul arhitecturii, medicinei și alchimiei. În această perioadă, caracterizată și în zilele noastre de realizări recunoscute ca adevărate 'minuni ale lumii', omenirea a realizat pasul decisiv în direcția abordarii analitice a fenomenelor reale. Bazate mai mult pe observația relațiilor între cauză și efect, aceste abordări au condus nu numai la crearea de noi materiale (e.g. hârtia/papirusul, praful de pușcă, medicamente), la apariția științelor naturii și a primelor aplicații bazate pe acestea (invențiile lui Leonardo da Vinci), ci au fost efectuați și primii pași în direcția abstractizării matematice. Datorită acestui mod de abordare a problemelor, asemănarea formei obiectului inițial și a celui rezultat nu mai este evidentă. Caracteristic este însă faptul că în cadrul tehnicilor RE, sistemul inițial este unul material, cunoștințele pluridisciplinare nefiind neapărat necesare.

2.4.1.3 III - Etapa/nivelul abstractizărilor matematice (asemănarea analitică)

Odată cu dezvoltarea matematicii analitice, gradul de abstractizare a sistemelor reale a ajuns la un nivel atât de superior încât tehnicile de RE sunt aplicabile nu numai asupra obiectelor reale, ci și asupra modelelor imateriale. Această etapă este caracterizată de necesitatea abordării științifice a problemelor prin utilizarea cunoștințelor pluridisciplinare. În această etapă, sistemele inițiale au părăsit chiar forma materială (e.g. analiza psihicului uman ducând la crearea modelelor matematice de rețele neuronale autoorganizabile).

De menționat este faptul că aceste etape, deși apărute în mod cronologic, pot fi privite și ca nivele ale tehnicilor de RE, deoarece fiecare din etapele superioare conține în sine etapa anterioară. Astfel, de exemplu tehnicile de RE bazate pe copierea formei în lut, plastelină, ceară sunt utilizate și în prezent.

Schematic, aceste niveluri pot fi reprezentate ca în Fig. 5.

Fig. 5: Etapele/nivelele RE

Îmbunătățirea permanentă a tehnicilor de RE a dus astfel la obținerea de sisteme noi sau modificate tot mai perfecționate. Ca o consecință a impactului acestui proces în viața economică și în special în vederea protejării inventatorilor și inovatorilor, a apărut necesitatea legiferării acestor tehnici, în special prin brevete de invenții/inovații, mărci și copyrights. Scopul principal sete astfel cel de a ușura persoanei sau instituției inovatoare returnarea investițiilor efectuate în vederea obținerii noului produs. Datorită tehnicilor RE existente, reingineria unui produs nemaifiind o problemă, aceste forme de legiferare au devenit indispensabile.

La începutul anilor 1970 au apărut și primele restricții legale referitoare la RE. Unele dintre aceste restricții au fost declarate explicit, altele părând a fi implicite în reglementări legale noi care nu amintesc noțiunea de RE, ca de exemplu în TRIPS (Trade-Related Aspects of Intelectual Property Rights)[5] din 1994 și EEA (Economic Espionage Act)[6] din 1996. TRIPS este un pact care obligă statele membre ale World Trade Organization (WTO) să protejeze secretele de afaceri (trade secrets), cu toate că nu precizază nici necesitatea și nici nu sancționează utilizarea tehnicilor de RE. Lipsa acestor precizări și deci și a protejării tehnicilor de RE a cauzat discuții, deoarece unele drepturi acordate în cadrul EEA implică în mod discutabil aplicarea unor anumite tehnici RE inițial considerate ca fiind legale.

Între 1970 și 1980, unele state, au intezis de exemplu formarea directă a corpurilor bărcilor și vapoarelor prin tehnici de RE[7].

La sfârșitul anilor '70 și începutul '80 industria producătoare de semiconductori a căutat și a obținut o lege de protecție a layoutului cipurilor care interzice clonarea acestora prin tehnici de RE.

Spre sfârșitul anilor '80 și începutul anilor '90 au apărut controverse în privința decompilării programelor soft (tot o formă de RE în domeniul ingineriei programării) punându-se problema legalității privind legile de copyright. Cu toate că instanțele judecătorești americane au considerat decompilarea ca legală în vederea obținerii unei mai bune interoperabilități, controversele au condus la întărirea și dezvoltarea regulilor de licențare, care în momentul actual interzic în general tehnicile de RE în acest domeniu. În această perioadă existau încă discuții, dacă decompilarea încalcă drepturile de patent[8]. În 1998, Congresul american a interzis prin lege aplicarea tehnicilor de RE a versiunilor digitale ale elementelor tehnice protejate prin copyright. Prin această lege este interzisă de asemenea programarea sau distribuirea de utilitare în vederea aplicării acestor tipuri de tehnici (cu exceptia unor circumstanțe foarte limitate), interzicând totodată și publicarea informațiilor obținute în urma utilizării tehnicilor RE legale[9].

O a doua adevărată revoluție în acest domeniu a apărut deci ca urmare a 'democratizării' procesorului și a 'exploziei' informatice. Prin impactul acestora asupra mediului industrial, s-a ajuns totodată la o exuberantă dezvoltare a tehnologiilor asistate de calculator atât în domeniile industriale tradiționale, în cele neconvenționale cât și în domenii care inițial nu păreau să aibe vre-odată legături interdependente cu tehnologiile informatice. Astfel tehnicile de RE s-au extins cu succes în domenii ca ingineria softului (Software Engineering), medicina, genetica, artele plastice, filmul sau arheologia.

Pe de altă parte însă, RE este parte constituentă opțională a Prototipării Rapide, permițând astfel crearea de noi sisteme sau ameliorarea (îmbunătățirea) celor existente într-un ritm tot mai rapid.

2.4.2 Exemple

Exemplele următoare se vor referi doar la domenii noi apărute în ultimii ani, deci nu vom mai aminti metodele, tehnicile sau tehnologiile consacrate, unele devenite în decurs timpului chiar ştiinţe ale naturii, cum ar fi fizica, chimia, mecanica.

Descifrarea codului genetic uman este tot un proces de RE, dat fiind faptul că prin înţelegerea modului de funcţionare a sistemului genetic, se obţin de la un sistem existent informaţii ce sunt abstractizate, iar în continuare adaptabile unor condiţii iniţial eventual inexistente. Crearea de noi specii de plante sau animale, precum şi adaptarea celor existente la anumite condiţii ca urmare a informaţiilor dobândite (deci prin utilizarea tehnicilor de Reengineering) este aşadar ca urmare a acestui proces, posibilă – iar în ultimul timp chiar, existentă.

Un alt exemplu, care în ultimul timp a dus la nenumărate controverse în special datorită frontului său larg de acţiune, este referitor la aplicarea tehnicilor de RE în domeniul ingineriei de soft (Software Engineering).

Prin analiza amănunţită a materialelor şi înţelegerea interacţiunilor la nivel molecular, au fost în ultimii ani create noi materiale, cu calităţi speciale şi specifice domeniilor în care sunt utilizate cum ar fi:

- materiale suprauşoare
- materiale supraconductive
- Wire Muscles (aliaje pe bază de Nitinol)

Pe de altă parte, tot bazat pe abstractizările amintite mai sus au apărut tehnologii noi cum este cazul nanotehnologiei.

2.4.3 Probleme

Problemele principale care apar sunt de natură economică şi legală.

2.4.3.1 Probleme de natură economică

Dat fiind faptul că prin aplicarea tehnicilor de RE timpul de îmbunătăţire a unui produs sau de realizare a unui nou produs este substanţial redus, perioada de timp necesară amortizării investiţiilor datorate cercetării şi dezvoltării nu mai este suficientă. Fără a scăpa din vedere importanţa ingineriei clasice (directe) şi nici a reingineriei, trebuie totuşi menţionat că ambele domenii pot fi direct legate şi deci profita de succesele RE.

Efectele economice ale RE depind de un numeroşi de factori, incluzând:

- scopul RE
- contextul industrial în care are loc
- costul procesului
- durata procesului
- dacă licenţarea este o alternativă viabilă
- modul în care inginerul aplică informaţia acumulată în cursul procesului de RE

Ţinând cont de faptul că celelalte aspecte sunt de natură legală, în continuare ne vom ocupa doar de faţetele economice ale RE întreprinse cu scopul dezvoltării unui produs competitiv.

În momentul de faţă problema greutăţilor economice ale unei firme datorate RE nu se pune, date fiind:

- avansul în procesul de producţie a produsului nou faţă de firmele concurente

– preţul relativ ridicat al RE

Pe de altă parte, chiar în cadrul aceleiaşi întreprinderi este uneori necesară utilizarea tehnicilor de RE, în general în cazul în care se urmăreşte îmbunătăţirea unui produs existent prin utilizarea noilor tehnici de analiză şi simulare asistate de calculator (FEM, ADAMS, VR, AVR, CAVE).

În vederea justificării aplicării în practică a RE, este necesară o analiză detaliată atât a ciclului de viaţă al produsului cât şi a eficienţei economice (cost/profit) rezultate.

2.4.3.2 Probleme de natură legală

Ca urmare a problemelor de natură economică a apărut necesităţii de protejare legală. În momentul actual procesul de definire a legilor cu valabilitate internaţională este doar la început. Marea majoritate a interdicţiilor referitoare la RE în momentul de faţă se referă la Software Engineering (SE). Motivul principal este probabil faptul că aparatele şi softul necesar RE în alte domenii în momentul de faţă sunt relativ scumpe şi/sau greu accesibile. În momentul în care aceste considerente se vor schimba drastic, este foarte probabil să se recurgă la adoptarea unor convenţii sau legi 'sănătoase', asemănătoare celor mai sus amintite referitor la formarea corpurilor de ambarcaţiuni (interzicerea remodelării utilizând numai tehnici de RE).

Ţelul principal este însă de a defini reguli legale care să protejeze sistemele complexe, în care a fost investită o cantitate mare de cunoştinţe şi informaţii împotriva clonării - care ar distruge economia de piaţă. Pe de altă parte este însă la fel de important să rămână un domeniu suficient de larg care să permită competitorilor noi pe piaţă o activitate inovatoare profitabilă, în caz contrar riscându-se o inhibare a factorului inovativ.

În general aplicarea tehnicilor de RE în domeniile tradiţionale de prelucrare este legală, cel puţin în momentul de faţă, la care ne vom referi în continuare (în cazul în care nu este altfel explicit specificat).

RE a fost tot timpul considerată ca o tehnică legală de acaparare a secretelor tehnologice, atât timp cât produsul supus RE a fost achiziţionat în mod legal, fair şi onest cum ar fi de exemplu cumpărarea pe piaţa liberă – fapt citat în „Official Comment on Sec. 1 of Uniform Trade Secrets Art". În 1993 însă, a avut loc o revizuire a descrierii de mai sus[10], prin care American Law Institute a reglementat scăpările conţinute, decretând că „... posesorul unui secret de afaceri nu deţine dreptul exclusiv de posesiune sau de utilizare a informaţiei secrete. Protecţia există doar în cazul achiziţionării incorecte sau publicării secretului informaţiei.", cum ar fi în cazul în care publicarea afectează negativ în mod explicit sau implicit convenţiile dintre parteneri sau în cazul în care metode neadecvate (forţare, înşelăciune) sunt utilizate în vederea obţinerii secretului.

O altă justificare a dreptului de utilizare legală a tehnicilor de RE rezultă din asemănarea sa cu cumpărarea şi utilizarea legală a produsului, care oferă posesorului toate drepturile de proprietate, inclusiv măsurarea, testarea, demontarea ş.a.m.d. Recompensa pentru timpul, banii şi energia investită în procesul de RE poate fi considerată în acest caz informaţia acaparată în timpul acestui proces. Astfel, vânzarea produsului pe piaţa liberă este considerată ca o publicare a informaţie pe care o conţine. Această publicare dedică astfel informaţia publicităţii, exceptând cazul în care creatorul şi-a patentat invenţia. Patentarea permite inventatorului o protecţie de până la 20 de ani a drepturilor exclusive de a produce, utiliza şi vinde invenţia, dar numai în schimbul publicării detaliilor semnificative ale invenţiei. Din această cauză, necesitatea RE asupra unui patent nu este posibilă – patentul fiind publicat, analiza modului în care subiectul patentului a fost creat devine inutilă.

2.4.4 Clasificarea RE legată de prelevarea de material

Deşi aplicabile în toate domeniile activităţilor umane, în continuare nu ne vom referi decât la tehnicile de RE legate de pelucrările prin prelevare de material.

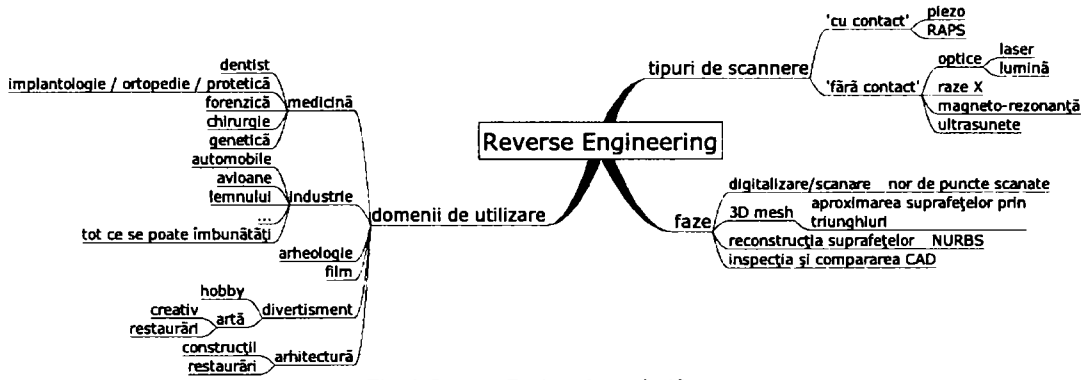


Fig. 6: Reverse Engineering - clasificare

O astfel de clasificare a RE după tipurile de scanere folosite, domeniile de utilizare cele mai răspândite și fazele tipice este prezentată în Fig. 6.

2.4.5 Fazele tipice ale procesului de RP

În perioada actuală, putem defini procesul de RP prin succesiunea de faze tipice prezentate în Fig. 7.

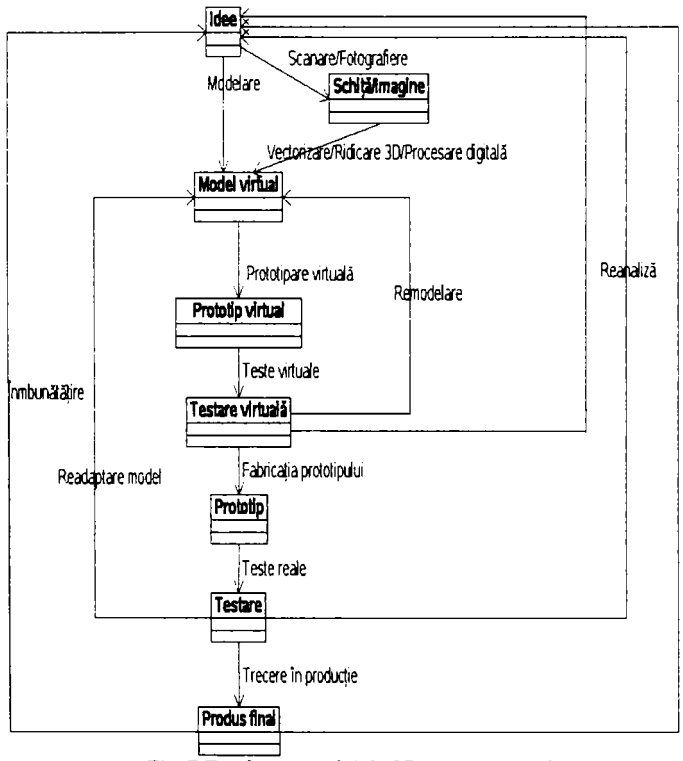


Fig. 7 Fazele procesului de RP în etapa actuală

2.4.5.1 *Scanarea/Digitalizarea*

Scanarea (digitalizarea) este tehnica principală utilizată în perioada actuală în vederea introducerii dimensiunilor modelului fizic în calculator, tehnica nefiind însă obligatorie deoarece, în general pentru modele simple, o măsurare a poziției punctelor caracteristice urmată de atribuirea lor manuală în programele de CAD este de asemenea posibilă – și deseori utilizată. În cazul obiectelor fizice reale complexe, modelarea fără utilizarea scanerelor și digitizoarelor specializate este de neconceput. Dat fiind însă faptul că scanarea implică baleierea prin puncte cât mai apropiate a suprafețelor exterioare și/sau interioare ale obiectului, rezultatul digitizării este un așa numit 'nor de puncte'.

2.4.5.2 *Aproximarea suprafețelor prin elemente poligonale*

Pasul specific următor este aproximarea suprafețelor prin generarea de elemente poligonale, în vederea obținerii suprafețelor. Singura condiție impusă este ca suprafețele obținute prin unirea anumitelor puncte să aparțină unei suprafețe plane elementare orientabile – în cazul suprafețelor scanate complexe, de cele mai multe ori fiind reprezentate de triunghiuri.

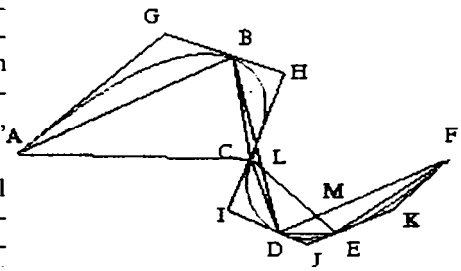


Fig. 8 Poligonalizare/Aproximarea prin triunghiuri

Deja în această fază, pentru cazul obiectelor simple, procesul de RE poate fi considerat ca fiind realizat. În anii trecuți, înainte de apariția modelelor matematice de aproximare a suprafețelor prin NURBS, modelul as fel ob inu era suficien de exac pentru a permite aplicarea analizei prin FEM, vizualizării și chiar a tehnicilor de Reengineering.

Dezavantajul care rezultă din simpla aproximare a suprafețelor prin elemente oligonale este recizia re rezentării. A a cum se poate observa din Fig. 8, aproximarea polinomială implică erori dimensionale.

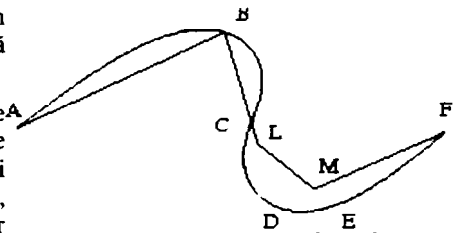


Fig. 9 Rezultatul aproximării poligonale prin triunghiuri

Astfe', pentru curba ată (de culoare verde), fiind digitalizate punctele A, B, C, D, E și F, avem determinate triunghiurile ABC, BCD, CDE și DEF. După cum se observă, în cazul zonei concave a curbei de digitalizat precum și a zonei de inflexiune, triunghiurile rezultate se suprapun, generând la rândul lor puncte de intersecție. Așadar, mulțimea de triunghiuri elementare necesară definirii digitale a conturului, este compusă din triunghiurile ABC, BCL, LCD, DML, MDE și EFM. Conturul final, definit prin punctele rezultate ale aproximării poligonale, ^s^^ ABCLMF. Î- ^^^t ^^^, ^^^^^^^^l d^ ^^li^^^^li^^^^ t^^- buie să fie suficient de inteligent în a alege poligonul definit prin punctele digitalizate și nu prin cele rezultate prin intersecții intermediare, aproximarea rezultată nefiind satisfăcătoare (Fig. 9).

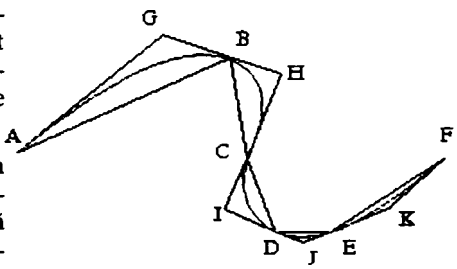


Fig. 10 Poziționarea conturului real în cadrul aproximării

Pe de altă parte, din Fig. 10 se poate observa că aproximarea prin punctele rezultate din intersecția tangentelor la curba inițială este, în ciuda erorilor totuși existente, mult mai apropiată de forma reală a obiectului de digitalizat, necesitând însă informații suplimentare ce nu pot fi definite doar prin poziția punctelor scanate. În cazul în care punctele de digitalizat au fost corect alese, curba de aproximat se va afla întotdeauna în interiorul unui triunghi determinat de punctele digitalizate și tangentele la curbă în aceste puncte. În cazul nostru deci, curba reală se află cu siguranță în interiorul triunghiurilor ABG, BCH, CID, DJE și EKF, după cum reiese din figura amintită anterior.

În cazul corpurilor real însă, triangularizarea are loc după un alt principiu, dat fiind faptul că suprafețele corpurilor scanate sunt delimitate de curbe închise. Datorită acestui fapt, triangularizarea se face sub condiția de neintersecție a liniilor dintre vertexuri (punctele de scanare utilizate ca vârfuri ale triunghiurilor) – procedeu cunoscut sub numele de maiaj sau tesselare. Modificând forma prezentată mai sus pentru cazul suprafețelor închise, aproximarea poligonală este definită de punctele scanate, iar una dintre modalitățile de tesselare este de asemenea prezentată în Fig. 11.

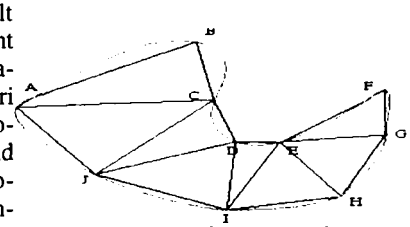


Fig. 11 Tesselarea/Maiajul

2.4.5.3 Aproximarea prin NURBS

Cea mai simplă și mai răspândită metodă de aproximare a curbei date prin informațiile dobândite din aproximarea poligonală și triangularizare este metoda de aproximare prin NURBS (Non-Uniform Rational B-Splines). Această metodă asigură intersecția curbei aproximate cu cea reală în punctele digitalizate prin definirea așa numitelor puncte de control.

Datorită avantajelor oferite, transformarea suprafețelor poligonalizate în suprafețe aproximate prin NURBS a devenit în procesul de RE inevitabilă, cu toate dezavantajele ce apar în partea de Reengineering – în principal datorate faptului că modificarea modelului astfel obținut este foarte dificilă, întrucât modificarea poziției oricărui punct atrage după sine modificarea mai mult sau mai puțin puternică a mai multor puncte din care este constituită curba sau suprafața NURBS.

Algoritmul prezentat în continuare[11] este bazat pe ISO 10303(STEP) cu intenția de a putea fi folosit în toate sistemele CAD. Următoare aspecte sunt direct legate de utilizarea acestor curbe:

- determinarea direcției optimale de construcțiilor bazate pe NURBS
- subdivizarea adaptivă a datelor NURBS pentru reprezentare geometrică cât mai exactă
- reprezentarea prin ray casting / ray tracing

Spre deosebire de aceste algoritme, cel prezentat în [11] este independent de orice sistem CAD și se bazează doar pe un fișier de intrare de tip STEP al modelelor de prelucrat. În principal sistemul urmărește eliminarea pasului de tesselare (STL) și transmiterea datelor CAD ale modelului direct la mașina de prelucrat.

NURBS a devenit standardul industrial de representare și design asistat de calculator a formelor libere, în special în construcția de mașini, construcții aerospațiale și navale, artă etc., deoarece metodele de stratificare (slicing) lucrează în acest caz direct pe reprezentarea matematică a modelului și nu pe aproximarea prin triunghiuri a acestuia.

Algoritmul propus a fost implementat în C++ și aplicat asupra 3 forme complexe de NURBS. Tabelul comparativ în vederea procesării, prezentat în [11] este:

Nume	*Format*	*Proprietăți*	*Dimensiune fișier de date*	*Timp de procesare*
Model 1	STEP	44 NURBS	164 KB	52 sec
	STL	52.868 triunghiuri	12.25 MB	4 sec
Model 2	STEP	102 NURBS	860 KB	71 sec
	STL	85.868 triunghiuri	25.50 MB	5 sec
Model 3	STEP	135 NURBS	1.09 MB	72 sec
	STL	74.120 triunghiuri	23.20 MB	3 sec
Model 4	STEP	78 NURBS	924 KB	72 sec
	STL	33.878 triunghiuri	8.55 MB	eșec

Tab. 1 Comparație NURBS/STL

În special pentru forme libere, prin utilizarea NURBS se pot elimina dezavantajele datorate utilizării tesselării (STL):

- tesselarea implică utilizarea suprafețelor triunghiulare, care în cele mai multe cazuri și în special pentru suprafețe cu formă liberă este nedorită, conducând la erori mari de prelucrare sau, ca de exemplu pentru Modelul 4 din Tab. 1, imposibilitarea stratificării
- cantitatea de date necesară reprezentării print STL este mult mai mare decât cea necesară prin NURBS. Chiar dacă stocarea datelor nu este un punct vital al prelucrării datelor, trebuie totuși menționat faptul că aceste date trebuie scrise, citite și interpretate, operațiile de intrare/ieșire sunt mult mai lente decât cele de calcul, iar cu creștere numărului de obiecte crește și șansa apariției erorilor
- foarte multe programe CAD refuză sau eșuează în timpul tesselării, iar implementarea unor algoritmi robuști și eficienți duce direct la mărirea timpului de execuție – care în moment este singurul avantaj al utilizării STL
- utilizarea NURBS permite stratificarea multimaterial și deci a modelelor heterogene
- prin NURBS se obține o abatere minimă a piesei fabricate comparativ cu modelul CAD
- timpul de procesare ridicat pentru NURBS poate fi micșorat prin implementarea unor algoritmi mai eficienți, rularea pe un calculator dotat cu un procesor mai rapid, calculatoare de performanță (HPC – High Performance Computing) sau prin paralelizarea calculelor și distribuirea lor pe mai multe calculatoare (GRID Computing, Clustering).

În cazul digitizării manuale 2D, suprafețele scanate fiind definite ca suprafețe delimitate de curbe plane, poligonalizarea (și eventual aproximarea prin NURBS) are loc de cele mai multe ori în timpul scanării, informațiile adiționale necesare aproximării fiind introduse de către operator (cerc, elipsă, linie dreaptă, etc.).

2.4.5.4 Integrarea CAD

După terminarea aproximării, pasul următor este cel de integrare în sistemul CAD utilizat. În cadrul acestui proces, modelul obținut este prelucrat în vederea eliminării eventualelor erori de aproximare, asignarea de atribute și caracteristici în vederea asigurării posibilităților de prezentare și analizare a comportamentului în cadrul aplicării metodelor de FEM sau a altor metode de simulare și verificare. Din acest punct de vedere, integrarea CAD este ultimul pas legat strict de RE, următoarele activități efectuate asupra modelului abstractizat aparținând domeniului de Reengineering.

2.4.5.5 Interfațarea cu alte sisteme (CAX, VRML, FEM, etc.)

Posibilitatea de interfațare a sistemelor CAD cu alte sisteme, în general CAM sau CAE, nu este nici strict necesară și nici strict legată de RE, oferă însă posibilități variate de analiză suplimentară a modelului generalizat, deci poate fi încadrată ca element de legătură cu alte sisteme specifice RE. Astfel, neclaritățile sau problemele legate de înțelegerea modului de funcționare a obiectului real pornind de la modelul abstractizat sau idealizat obținut în urma procesului de RE, pot fi prezentate în cadrul unor sisteme de CSCW (Computer Supported Colaborative Working) sau VR (Virtual Reality)/AVR (Augmented Virtual Reality), discuțiile rezultate în urma prezentării ducând în general la o înțelegere mai profundă atât a modului de funcționare a obiectului / sistemului real, cât și a posibilităților de îmbunătățire a acestuia (să nu uităm faptul că RE este în cele mai multe cazuri procesul inițial necesar în vederea aplicării tehnicilor de Reengineering).

O altă interfață foarte importantă și care câștigă în ultimul timp tot mai mult teren este cea legată de sistemele distribuite de calcul. Datorită metodelor numerice tot mai sofisticate, puterea de calcul necesară rulării programelor de analiză și simulare este tot mai mare. Nu numai datele de intrare în analize sau simulări prezintă din acest motiv probleme tot mai ridicate în vederea prezenței și arhivării lor, ci și mai ales datele de ieșire, rezultatele simulărilor efectuate. Variabila însă cu cea mai mare pondere este însă în majoritatea cazurilor timpul de execuție al simulării.

Cu toate că în ultimii ani tehnica de calcul a evoluat exploziv, deși în domeniul cercetării sistemele de calcul cu mii de procesoare, memorie externă de ordinul Terabyților și internă de sute de Gigabyte nu mai sunt rarități, cerințele referitoare la resurse sunt tot timpul în urma situației reale. Din acest motiv, în ultimul timp inițiativele legate de GRID computing câștigă tot mai mult teren. Majoritatea bibliotecilor existente și a algoritmilor existenți permit distribuirea programelor interactiv sau automat pe mai multe calculatoare, în principal cu scopul de a reduce timpul de rulare necesar obținerii rezultatelor.

Dat fiind faptul că distribuirea este parte componentă atât a sistemelor de operare existente la ora actuală (sau în cel mai rău caz prin instalarea utilitarelor adecvate) și totodată limbajele de programare moderne ușurează enorm această facilitate, atât prin utilizarea optimală procesoarelor existente pe un singur sistem de calcul cât și prin distribuția urmată de sincronizarea rezultatelor pe sisteme de calcul diferite.

Din acest motiv, unul dintre premizele aplicațiilor implementate ca exemplu în cazul lucrării de față este portabilitatea acestora atât din punct de vedere al limbajului adoptat, a interfețelor cu sistemul de operare și bineînțeles a interfeței cu utilizatorul.

3 Prototiparea rapidă subtractivă (Subtractive Rapid Prototyping)

3.1 Hardware

3.1.1 Roland DG

SRP [12] a fost initial introdusă și înregistrată ca marcă de către firma Roland DG, o firmă japoneză producătoare de instrumente muzicale și echipamente periferice de calculatoare, care în momentul de față produce, livrează și oferă pe pieţele de inginerie și producție echipamente și soluții în domeniul frezării și scanării, în special legate de prototiparea rapidă. Firma este o fiică a cunoscutei firmei Roland Corporation, firmă fondată în 1972 și cunoscută în lumea întreagă prin instrumentele muzicale produse. În anul 1982, prin necesitatea datorată desenării cît mai exacte a undelor sonore generate de sintetizoarele de sunet, a fost fondată firma Roland DG (Digital Group), producătoare a primului plotter DXY-100. Bazat pe experiența acumulată în domeniul plotterelor, firma s-a extins exloziv și în domeniul imprimării colore, produselor de tăiere – în special în domeniul grafic. O dezvoltare similară au suferit-o nevoile firmei în domeniul liniilor proprii de așchiere. Folosindu-se de cunoștințele acumulate în domeniul poziționării X-Y a plotterelor, mașinilor de tăiat și imprimantelor, firma Roland a reușit cu succes adăugarea celei de-a treia coordonate pentru crearea liniei de freze CAMM-3. În anul 2002, investiția anuală a firmei în domeniul 3D crescuse la 36%. În momentul de față, liniile interne de fabricație ale firmei sunt în întregime compuse din liniile MDX-500 și MDX-600 iar partea de masurători este făcută începând din 2004 de o mașină de tip CMX-500, bineînțeles produs al firmei. În ultimii ani, liniile MDX-650 au fost sistematic îmbunătățite, prin adăugarea unei a patra axe (rotative) și a unui sistem de schimbare automată a sculelor (ATC – Automatic Tool Change).

În ultimi ani, firma Roland și-a extins liniile de fabricație oferind produse atât de uz profesional, cât și personal, dată fiind apariția unei noi piețe de desfacere alcătuită din hobby-iști, sculptori, bijutieri și cadre didactice. Aparaturile și echipamentele livrabile (MDX-15 și MDX-20) oferă o utilizare simplă și directă, asemănătoare mai mult cu utilizarea unei imprimante decât a unei mașini CNC. Împreună cu Modela Player și VisualMill Basic – pachete de CAM – sistemele oferă la un preț foarte convenabil posibilitatea unei utilizări rapide și simple. Ambele sisteme incorporează un scanner de contact, care împreună cu programul Dr. Picza) permit aplicarea metodelor de Reverse Engineering la un preț rezonabil. În afara acestor scannere, firma oferă și alte două scannere de contact dedicate, PIX-4 și PIX-30. Aceste scannere incorporează o altă tehnologie a firmei, Roland Active Piezo Sensor (RAPS), care elimină deficiențele scannerelor 3D de contact, permițând scanarea tuturor tipurilor de materiale – moi sau tari, opace sau transparente.

În domeniul profesional, firma Roland oferă două freze CNC – MDX-500 și MDX-650 – și un scanner laser – LPX-250. Prețul de start al frezelor este cca. $20.000, iar a scannerului de cca. $10.000.

Avantajele lor, comparativ cu alte echipamente de pe piață sunt:

1. sisteme uscate
2. alimentare bifazată
3. două limbaje de interfață – RML și G-code care permit o adaptare ușoară atât pentru inginerii de design cât și pentru specialiștii în frezare CNC
4. MDX-650 permite atașarea opțională ATC și a unei

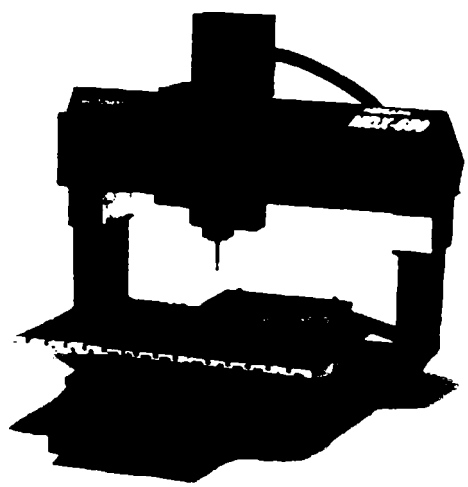


Fig. 12 Roland MDX-650

a patra axe (de rotație)

5. Scannerul este livrat cu o variantă OEM a programului PixForm – un subset al programului RapidForm al firmei Inus Technologies – și permite exportarea datelor achiziționate în limbaje standard (STL, IGES). Dotat cu o masă rotativă, scannerul permite scanarea pieselor într-o singură fază

O comparație a componentelor MDX-15/MDX-20 amintite mai sus este prezentată în Tab. 2:

	MDX 15	***MDX 20***
Motor	10W, 6500 rpm, 0,1 – 15 mm/sec	
Rezoluție mecanică	0,0625 mm	
Tip	ac și sensor piezo activ	
Mărimea maximă de scanare	152 x 101 x 60	203 x 152 x 60
Greutatea maximă de scanare	500 g	1000 g
Mărime	426 x 280 x 305	477 x 281 x 305
Greutate	9,6 kg	13,7 kg
Rezoluție de scanare	X/Y 0,05 – 5 mm, Z -- 0,025 mm	
Viteza de scanare	4 – 15 mm/sec	
Formate de export	DXF, STL, 3DMF, VRML	
Interfețe	Serial	
Componente livrate	Alimentare,Instrucțiuni de folosire, Cap de scanare, Cap de frezare	
Software	Dr.Picza,Modela Player	

Tab. 2 Echipamente de Frezare și scanare SRP ale firmei Roland

3.1.2 ISEL Group

ISEL este o asociație de firme producătoare de sisteme NC i CNC.

Dintre cele mai cunoscute sisteme de decupare (Router) putem aminti linia DaVinciSteper[13], a firmei TechnoISEL care, datorită precizie prelucrării și prețului ridicat, se pretează în primul rând pentru producție dar și pentru prototipări la care precizia de prelucrare este foarte importantă (Tab. 3). De asemenea firma este cunoscută datorită numeroaselor aplicații în domeniul prelucrărilor pe bază de lemn, plastic și gravuri. Astfel numeroase firme se folosesc de aceste sistemele în special în vederea restaurărilor mobilei clasice, a îmbunătățirii calității instrumentelor , p ș g . Un n n n n care firma s-a evidențiat în ultimii ani, este cel de Reverse Engineering în domeniul Automotiv prin noul sistem de scanare cu laser al firmei NVision (Dallas, Texas), un sistem Laser portabil, care poate fi montat pe o mașină TechnoCNC, asigurând astfel r p .

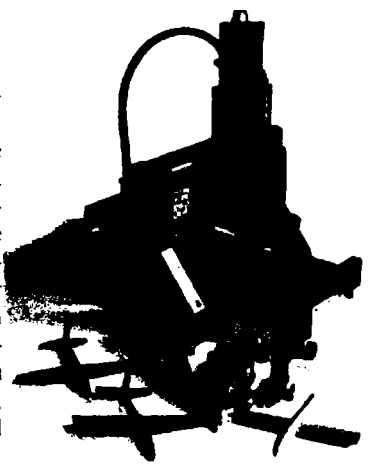


Fig. 13 TechnoISEL

Dată fiind precizia mare atât de scanare câ și de prelucrare, sistemele sunt folosite totodată și în industria constructoare de avioane și navală (atât în domeniul civil, militar și de agrement).

	DaVinci1	DaVinci2	DaVinci3
Mărimea maximă de prelucrare	200x200x125	200x240x125	250x300x90
Mărime	850x650x650	950x650x650	1040x700x770
Greutate	32 kg	36 kg	43 kg
Rezoluție de prelucrare (repetabilitate)	<0,01 mm (<0,004 mm/axă)		
Viteza de prelucrare	0,3 – 60 mm		
Interfețe	RS232		
Componente livrate	2 axe liniare și 2 circulare		
Software	PAL		
Preț	în funcție de componente		

Tab. 3 Date tehnice Techno ISEL

Din nomenclatorul de masini NC și CNC ale firmei ISELAutomation putem aminti :

	ICV 4030	ICP 4030	ICP 3020	ICP 2015
Frezare				
Motor	500-1050W, 11000 - 2500 rpm			
Rezoluție mecanică	0,1 mm			
Mărimea maximă de prelucrare	600 x 375 x 170 mm	400 x 300 x 140	300 x 200 x 90	200 x 150 x 90
Mărime	771 x 834 x 1250	780 x 850 x 810	610 x 650 x 715	535 x 600 x 690
Greutate	130 kg	120 kg	102 kg	95 kg
Viteza de prelucrare	80 mm/sec	60 mm/sec		
Interfețe	EPP, 4 axă servo	RS232		
Componente livrate	Componentele pot fi comandate separat			
Software	isy-CAM 2.5Lite (livrare), isy-CAD 3.0, isy-CAM 3.0			
Preț	în funcție de componente			

Tab. 4 Date tehnice ISEL Automation

3.1.3 Sherline/TheCoolTool

Deși paleta oferită de firma TheCoolTool din Modling/Austria este foarte vastă, în cadrul lucrării de față au fost utilizate liniile UNI-Turn și UNI-Mill. Sisteme similare sunt oferite pe piața americană de către firma Sherline, sistemele oferite de firma TheCoolTool fiind adaptate pentru piața europeană. Date lor tehnice sunt amintite în Tab. 4:

	UNI-Mill Deluxe		UNI-Turn	
Tip	5410	2010	2000	4300
Spațiu de lucru	203mm · 8,00"	229mm · 9,00"	-	-
Deplasare pe axa X	228mm· 9,00"	229mm · 9,00"	110 mm · 4,25"	110 mm · 4,25"
Deplasare pe axa Y	127mm · 5,00"	178mm · 7,00"	-	-
Deplasare pe axa Z	159mm · 6,25"	137mm · 5,38"	-	-

	UNI-Mill Deluxe		UNI-Turn	
Lățime	356mm · 14,00"	565mm · 22,25"	190mm · 7,50"	220mm · 8,75"
Lățime totala	381mm · 15,00"	381mm · 15,00"	610mm · 24,00"	820mm · 32,25"
Înălțime totala	527mm · 20,75"	568mm · 23,38"	150mm · 6,00"	200mm · 8,00"
Mărimea mesei	70x330mm	70x330mm	-	-
Înălțimea centrului peste sanie	-	-	45mm · 1,75"	45mm · 1,75"
Înălțimea centrului	-	-	90mm · 3,50"	90mm · 3,50"
Distanța între vârfuri	-	-	200mm · 8,00"	430mm · 17,00"
Spindelbohrung · hole through spindle	10mm · 0,405"	10mm · 0,405"	10mm · 0,405"	10mm · 0,405"
Spndelgewinde · spindle nose thread	3/4" - 16 T.P.I.	3/4" - 16 T.P.I.	3/4" - 16 T.P.I.	3/4" - 16 T.P.I.
Spindel Konus · spindle nose taper	#1 Morse	#1 Morse	#1 Morse	#1 Morse
Reitstock · travel of tailstock	-	-	38mm · 1,49"	38mm · 1,49"
Reitstock Konus · taper of tailstock spindle	-	-	#0 Morse	#0 Morse
Drehbare Hauptspindel · protractor graduations	0° - 45°	0° - 45°	0° - 45°	0° - 45°
Gradații	0,01mm · 0,001"	0,01mm · 0,001"	0,01mm · 0,001"	0,01mm · 0,001"
Motor	100-240V 375-500W	100-240V 375-500W	100-240V 375-500W	100-240V 375-500W
Control electronic al vitezei de rotație	70 - 2.800 rpm	70 - 2.800 rpm	70 - 2.800 rpm	70 - 2.800 rpm
Greutate	16,3kg · 36lb	17,2kg · 38lb	10,9kg · 24lb	13,6kg · 30lb

Tab. 5: Date tehnice UNI-Mill/UNI-Turn

Dintre avantajele de bază ale acestor utilaje putem aminti:

- preț redus
- precizie ridicată
- alimentare bifazată
- sisteme uscate
- interfață Windows și Linux
- 4 axe de prelucrare

4 Software

Una dintre elementele esențiale ale prototipării rapide, fără de care sistemele actuale n-ar fi azi la nivelul la care sunt, este partea de soft. Deși în continuare vom aminti cele mai cunoscute și mai puternice programe utilizabile la ora actuală, ne vom folosi de ele doar în vederea comparării lor cu sistemele de tip Open Source actuale. Este evident că aceasta comparație nu poate fi deocamdată favorabilă sistemelor OpenSource, dat fiind faptul că vom compara sisteme profesionale comercial existente și dezvoltate în decursul multor decenii, cu sisteme care abia acum încep să-și facă apariția pe piață. Cu toate acestea, speranța ca aceste sisteme să devină cu timpul tot mai accesibile este pe zi ce trece tot mai mare, datorită în special naturii și modului de dezvoltare a sistemelor OpenSource, amintite în cadrul lucrarii în capitolul respectiv.

4.1 CAD

Autodesk CAD (AutoCAD) - Probabil cea mai cunoscută firmă de CAD, datorită nu numai vechimii sale pe piață ci și a posibilităților de modelare, vizualizare și simulare pe care le oferă.

CATIA - Produs al firmei IBM, cel mai cunoscut și utilizat sistem de CAD sub sistemul de operare UNIX. La ora actuală numărul firmelor utilizatoare este de peste 24.000 și peste 180.000 de persoane care utilizează produsul. Împreună cu „CATIA V5 STL Rapid Prototyping 2" al firmei Dassault Systems sistemul poate fi folosit și în prototiparea rapidă.

hyperCAD - Considerat ca „The best 3D software" de către companiile principale ale pieței internaționale, datorită faptului că print utilizarea produsului s-au realizat creșteri de productivitate de până la 90%. Programul este foarte intuitiv, și ueci ușor de utilizat, iar modulul său de bază (think3 kernel) permite o rapidă modelare, editare și corectare a datelor. Împreună cu o nouă tehnologie, denumită 'Smart Objects', care permite salvarea și reutilizarea de componente în/din biblioteci, precum și a elementelor inteligente de design și concept, concomitent cu adaptarea automată la geometrii și topologii variate, produsul permite eliminarea acțiunilor redundante sau de rutină și chiar completarea inteligentă a suprafețelor, devenind unul dintre cele mai remarcabile programe de CAD actuale.

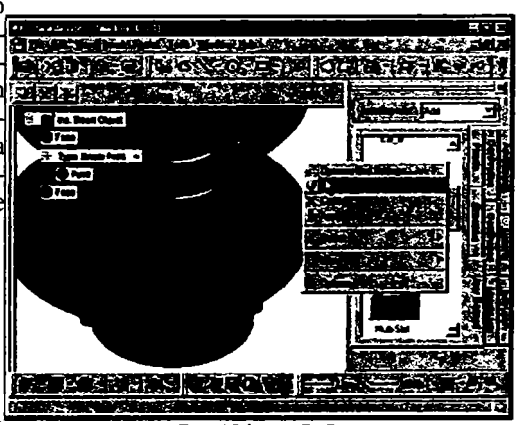


Fig. 15 hyperCAD

4.2 CAE / CAM / FEM

4.2.1 Prelucrare

ModelaP ay r 4 - Face p rte din p_ h tul _ liv_ al scannerelor firmei Roland, și reprezintă deci softul de bază al mașinilor de frezat ale firmei.

hyperMILL - Produs al firmei OPEN MIND Technologies AG, este unul dintre cele mai performante programe CAM cu posibilitatea importării de date CAD din majoritatea programelor cunoscute. Una dintre facilitățile cele mai importante este așa numita Feature-Technologie, prin care programul recunoaște de exemplu automat orificiile din piesa de prelucrat și generează codul NC de găurire, o programare separată a orificiilor de același tip nemaifiind necesară.

Posibilitatea de simulare grafică a prelucrării este bineînțeles integrată.

Optimizarea folosirii, alegerii și deplasării sculelor face parte de asemenea din funcționalitățile produsului, evitându-se astfel complet coliziunile și reducându-se drumurile fără așchiere ale sculei.

Alegerea suprafețelor de frezat este aleasă intuitiv și interactiv prin simpla selectare a suprafeței (sau suprafețelor).

Numeroase strategii de frezare HSC fac parte din ansamblul de rutine integrate.

Suport pentru mașini de prelucrat cu până la 5 axe. În cazul în care mașina de prelucrat nu poate acționa simultan toate cele 5 axe, programul preia automat repartizarea secvențelor de lucru.

Alte sisteme CAD/CAM - Liste sistemelor CAD și CAM nu este în nici un caz exhaustivă și se extinde foarte rapid, fapt pentru care, o altă listă – ordonată alfabetic, dar din aceleași motive la fel de incompletă – este prezentată în Tab. 6.

Produs	*Producător*	*CAD/ CAM sau CAM*	*Platforme*	*# Axe*	*Formate de import*	*Tip de Modelare*
Advanced System 3000	CamSoft Corp.	CAD/ CAM	UNIX	2, 3, 4 & 5	IGES, DXF, STL, SLA, Gerber, HPGL, CadKey & APT,CATIA CL, Excellon, Gerber	Nurbs
AlphaCAM	Licom Systems	CAD/ CAM	Windows 95, 98, 2000, NT	2,3,4 & 5	DXF, DWG, IGES, STEP, Parasolid, Rhino, CADL, ANVIL, BMP, TIFF, STL	Nurbs
ArtCAM Pro	Delcam plc.	CAD/ CAM	Windows 2000, XP	3 & 4	3D studio, 3D DXF, STL, IGES, BMP, TIF, EPS, AI, 2D DXF, 3D Laser, Renishaw & G-code	3D Relief (Proprietar)
Deskproto V3	Delft Spline Systems	CAM	Win 9x/2x/NT/ XP	3 & 4	STL, DXF (polygon data only), VRML	N/A doar pentru CAM
DeskProto Lite	Delft Spline Systems	CAM	Win 9x/2x/NT/ XP	3	STL, DXF (polygon data only), VRML	N/A doar pentru CAM
MillWizard	Delcam plc.	CAM	Windows 95, 98, 2000, NT 4.0	3	3D studio, 3D DXF	N/A doar pentru CAM
PowerMILL	Delcam plc.	CAM	Windows XP	2, 3 ,4 & 5	IGES, VDA, STL, STEP, DXF, DWG, Solid Works, Solid Edge, UG, ProE, Catia, ACIS, ParaSolid, Rhino, Cimtron, SDRC	N/A doar pentru CAM
RAMS Gold	RAMS Software, Inc	CAM	WINX. NT, 2000, XP	2, 3 & 4	EPS, DXF, 3D-DXF, STL, IGES, VDA, 3dm, SLA	N/A doar pentru CAM

Produs	*Producător*	*CAD/ CAM sau CAM*	*Platforme*	*# Axe*	*Formate de import*	*Tip de Modelare*
SimpleNC	Graphic Products North America	CAM	IBM	2 & 3	IGES, STL, RHINO, DXF, Cadceus, FRESDAM	Nurbs
StlWork	IMService	CAM	Wintel	3	STL, DXF, Vrml, 3Ds & others	N/A doar pentru CAM
Surfcam	Surfware	CAD/ CAM	Windows	2,3,4 & 5	IGES,DXF, DWG, Parasolid, Solid Edge, Solid Works, SAT, VDA, CATIA (up to version 5), Inventor, ProE, Step, UG	Nurbs, Suprafețe, Wireframe
Vector-Cam	Centriforce	CAD/ CAM	Wintel	2, 2.5, 3 & 4	DXF	Wireframe, Suprafețe
VisualMill v3.0 Basic	MecSoft, Inc.	CAM	PC Windows	2.5 & 3		N/A doar pentru CAM
VisualMill v3.0	MecSoft, Inc.	CAM	PC Windows	2.5, 3 & 4	IGES, Parasolides, Rhino, DXF, DWG, STL, SLA, VRML, VDA/FS, SolidWorks & SolidEdge	N/A doar pentru CAM

Tab. 6 Sisteme CAD/CAM

4.2.2 Scanare

Similar cu list.. p.og.....elo. CAD.CAM, ..st.. acestor programe este în continuă creştere, motiv pentru care vom aminti doar pe cele mai răspândite în momentul actual.

Dr.Picza - Programul este în pachetul de livrare al scannerelor firmei Roland DGA

Rhinoceros 3D - Este unul dintre cele mai performan e programe e scanare/ g t zare ş pre ucrare a datelor obţinute prin aceste metode.

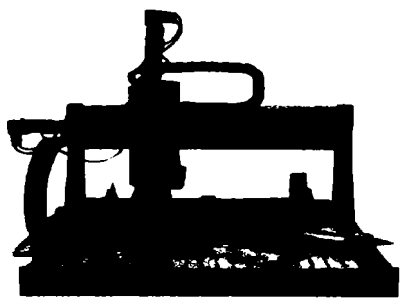
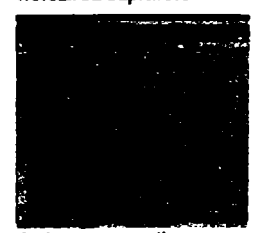


Fig. 17 Mastercam

Introducerea punctelor prin scanare sau tastatură

Crearea de wire frame şi netezirea suprafetei

Og.indire, ro.ire, editare şi al e m dificări



Ap.icarea texturilor, r nd ring...

Fig. 16 Rhinoceros 3D

Techno-ISEL Mastercam - Softul maşinii permite atât scanarea cât şi prelucrarea obiectelor.

Este un program utilizat în special pentru maşinile grupului ISEL, atât pentru CAD cât şi CAM, dar din documentaţie nu reiese în mod explicit nici numărul de axe ale de prelucrare suportate şi nici sistemele de operare sub care programul lucrează, motiv pentru care n-a fost introdus în Tab. 6.

4.2.3 VP (Virtual Prototyping)

Datorită dezvoltării rapide a softului pe de o parte şi a necesităţii reducerii costurilor de prototipare – de menţionat este faptul că în anul 2000 costurile rezultate în urma prototipării rapide numai USA, au fost estimate la cca. 10 miliarde de dolari - prototiparea rapidă a ajuns la un nivel atât de ridicat, încât termenul de Prototipare Virtuală (VP – Virtual Prototyping) câştigă din ce în ce mai mult teren[14], fiind considerată ca extensia logică a CAD şi CAE. Prin aceste sisteme, partea de test (statică şi/sau dinamică) a prototipurilor este efectuată în primul rând prin simulare şi de abia după succesul acestor teste se trece la prototiparea fizică. Dintre sistemele cele mai performante în perioada actuală putem aminti:

ADAMS – Automated Dynamic Analysis of Mechanical Systems

Parte a produsului MSC-Office al firmei MSC-Software, compus din următoarele module[15]:

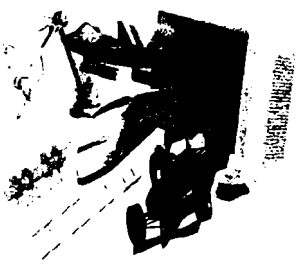


Fig. 18 ADAMS

ADAMS/3D Road - Permite crearea 3D a drumurilor, parcărilor, pistelor de curse, autobenzi, etc. pentru vehicolele virtuale.

Fig. 19 ADAMS-Road

ADAMS/Aircraft - Vine în ajutorul inginerilor prin crearea de prototipuri virtuale de avioane care pot fi testate, îmbunătăţite şi optimizate pe computer înainte modelării primului prototip real.

Fig. 20 ADAMS-Aircraft

ADAMS/AutoFlex - Puternice capabilităţi de analiza flexibilităţii corurilor solide fără a necesita utilizarea unor programe complexe de analiză prin elemente finite

Fig. 21 ADAMS-Autoflex

ADAMS/Car - Generarea unui prototip virtual complet al unui vehicol, combinând acurateţea modelului matematic al subsistemelor şasiului, a motorului şi a sistemului de direcţie, sistemului de control şi a caroseriei.

Fig. 22 ADAMS-Car

ADAMS/Chassis este un produs 'automotive' bazat pe simulare, care oferă o bibliotecă extinsă de mostre ('templates') de modelare şi analiză specifice standardelor industriale.

Fig. 23 ADAMS-Chassis

ADAMS/Controls ajută la integrarea simulării mişcării şi simulării sistemelor de control.

Fig. 24 ADAMS-Controls

ADAMS/Driveline permite crearea şi simularea prototipurilor v[illegible] d[illegible] c[illegible] ş[illegible] vehiculelor virtuale, a durabilităţii, vibraţiilor şi controlului lor.

Fig. 25 ADAMS-Driveline

ADAMS/Driver p[illegible] t cond[illegible] cerea simulată vehicolului virtual modelat, fără costurile şi întârzierile specifice prototipării fizice.

Fig. 26 ADAMS-Driver

ADAMS/Durability permite utilizarea modelelor existente în biblioteca MSC.ADAMS în vederea simulării testelor de oboseală.

Fig. 27 ADAMS-Durability

ADAMS/Engine powered by FEV este o suită de utilitare soft care o[illegible]er[illegible] funcţ[illegible]e crearea ş[illegible] s[illegible] mularea prototipurilor virtuale de motoare complexe.

Fig. 28 ADAMS-FEV

ADAMS/Exchange oferă interfeţe în ambele sensuri (two-way), bazate pe standarde industriale de transfer de date între linia de [illegible]roduse MSC.ADAMS şi cele mai cunoscute programe de CAD/CAM/CAE.

Fig. 29 ADAMS-Exchange

ADAMS/Flex permite analiza studiului influenţei interacţiunii părţilor [illegible]l[illegible]x[illegible]b[illegible]l[illegible] î[illegible]r[illegible]u[illegible] [illegible]i[illegible]t[illegible] [illegible]c[illegible]nic co[illegible]pl[illegible]x.

Fig. 30 ADAMS-Flex

ADAMS/Insight permite aplicarea de [illegible]od[illegible]l[illegible] [illegible]xp[illegible]rim[illegible]ntale asupra oricărei simul[illegible]ri, permiţând astfel o mai bună cuantificare a rezu[illegible]tate[illegible]or, ş[illegible] o ma[illegible] precisă apreciere a compromiselor de design aplicate.

Fig. 31 ADAMS-Insight

ADAMS/L[illegible]near ajută la liniarizarea [illegible]au simpl[illegible]f[illegible]r[illegible] [illegible]au-ţ[illegible]l[illegible] -el[illegible]i[illegible] d[illegible] cadrul MSC.ADAMS în vederea verificării fidelităţii modelelor .

Fig. 32 ADAMS-Linear

ADAMS/PostProcessor st interfaţa grafică principală de vizualizare a rezultatelor simulării prin MSC.ADAMS.

Fig. 33 ADAMS-PostProcessor

Cu ajutorul **ADAMS/Rail**, pot fi realizate modele complete de vehicole feroviare şi simularea realistică a comportării, îmbunătăţirii şi optimizării performanţelor înaintea rulării testelor fizice.

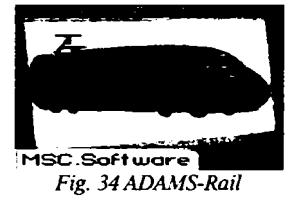


Fig. 34 ADAMS-Rail

ADAMS/Solver este o bibliotecă de analiză numerică rapidă și robustă, specializată în formularea și integrarea ecuațiilor ce guvernează simularea sistemelor mecanice.

Fig. 35 ADAMS-Solver

ADAMS/Tire - Simularea precisă a vehicolelor virtuale.

Fig. 36 ADAMS-Tire

ADAMS/Vibration permite studiul vibrațiilor din cadrul modelelor MSC.ADAMS utilizând analiza domeniilor de frecvențe în vederea identificării, izolării și îmbunătățirii nivelului de vibrații.

Fig. 37 ADAMS-Vibration

ADAMS/View - oferă posibilități superioare de modelare și vizualizare pentru simulări mecanice într-un cadru general.

Fig. 38 ADAMS-View

UGS[16] - Cu toate că paleta de produse ale firmei UGS este imensă, acoperind toate părțile ingineriei integrate, ca de exemplu module de Knowledge-Management, Teamcenter, Process-Planning, Resource Management, optimizarea întreprinderii și fabricației ne vom referi în continuare doar la cele legate de prototiparea rapidă prin prelevare de material.

SolidWorks - Sistem folosit în ingineria mecanică, design industrial, robotică, construcții aerospațiale, tehnologii de producție, sisteme automotive având module integrate de design, prezentare, animație și ca module opționale de analiză prin elemente finite, cinematică, dinamica fluidelor și termotehnică.

Una dintre cele mai importante element este însă comunicarea bidirecțională cu ADAMS.

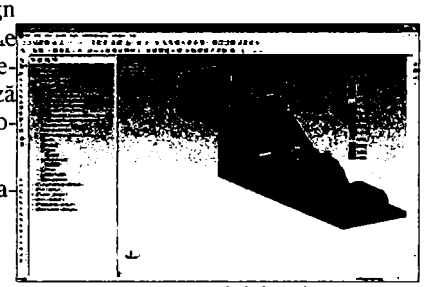


Fig. 39 SolidWorks

NX

I-DEAS Design Tools

Master Modeler

Sistem performant de CAD 3D având o interfață adaptată în vederea unei performanțe optimale a modelării structurilor mecanice complexe în mod intuitiv, cu posibilitatea interfațării cu sistemele NC. De asemenea este partea de bază a sistemelor I-DEAS, utilizabil nu numai ca interfață CAD ci și pentru crearea geometriilor specifice în vederea utilizării lor în alte module I-DEAS, ca de exemplu analiza și modelarea prin elemente finite, schițarea de componente sau prelucrarea lor. Permițând modelarea în echipă, asamblarea componentelor, simularea și [illegible] de job-uri [illegible] prelucrării NC, este o componentă importantă în ingineria simultană.

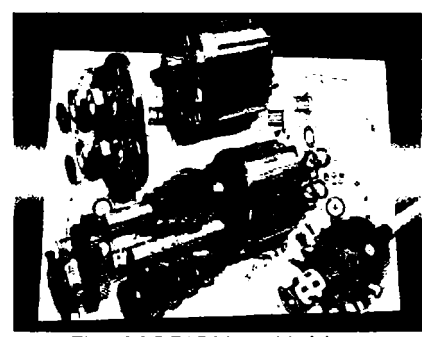


Fig. 40 I-DEAS MasterModeler

Master Notation

Modul I-DEAS specializat în realizarea desenelor de execuție.

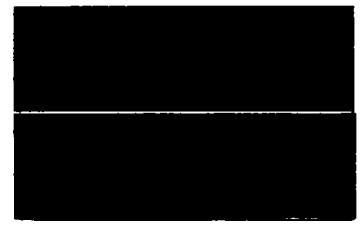


Fig. 41 I-DEAS Master Notation

Assembly Set

Este un modul I-DEAS cuprinzând numeroase ansambluri și subansambluri definite și incorporate ca bibliotecă și utilizabile în componente proprii.

Fig. 42 I-DEAS Assembly Set

Artisan Drafting

Permite generarea desenelor tehnice și schițelor în echipă, în special a componentelor solide, deci prelucrabile prin prelevare de material.

Fig. 43 I-DEAS Artisan Drafting

Mechanism Design

Permite simularea cinematicii corpurilor mecanice modelate

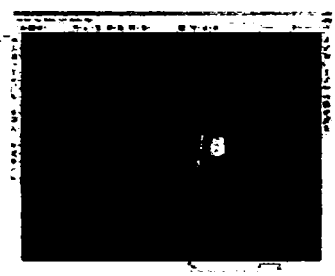


Fig. 44 I-DEAS Mechanism Design

Open I-deas

Ansamblu de utilitare 'Open Source'.

I-deas Translators

O funcționalitate foarte importantă a sistemului I-DEAS este posibilitatea de interfațare cu alte sisteme CAD, CAM, CAE de pe piață. Din modulele putem aminit:

- Stand Alone from CATIA
- STEP
- JAMA-IS
- CADAM to/from I-deas Master Drafting
- CADDS Solids
- Pro/ENGINEER Solids
- PCB.xchange and PCB.modeler
- PDGS Direct
- Unigraphics
- AutoCAD to/from I-deas Master Drafting

NX MasterFEM - Unul dintre avantajele prelucrării prin NX este integrarea simulării și verificării, ceea ce permite programatorilor să verifice traseele sculelor în cadrul sesiunii de prelucrare NC. Această capabilitate previne necesitatea de copiere a datelor în programe ex- [illegible] teci de componente, cât și scule, dispozitive de fixare și modele ale mașinilor de prelucrare.

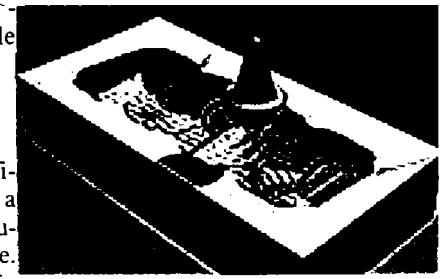


Fig. 45 I-DEAS Toolpath Verification

Toolpath verification

Softul este specializat în optimizarea traseelor sculelor și verificarea posibilelor erori, NX include posibilități de vizualizare a proceselor de prelevare de material și simularea deplasării sculei, verificând totodată prelevarea de material și coliziunile. Programul ajută de asemenea operatorul sau programatorul în depistarea eventualelor erori și optimizarea traseului sculei înainte de începerea reala a procesului de așchiere, permițând validarea digitală a programului NC prin simularea prelucrării. Astfel, verificarea programului este posibilă instantaneu, fără a fi necesară asteptarea generării complete a traseului sculei.

Machine tool simulation

NX machine tool simulation verifică programele NC printr-o simulare în contextul prelucrării pe masini-unelte. Astfel de simulări pot fi extrem de folositoare în cazul prelucrării pe mașini complexe, cu mai multe axe, cu capete multiple ca în cazul centrelor de frezare și strunjire. O versiune avansată a programul permite chiar definirea de modele de mașini unelte prin intemediul unui modul special, numit 'too˙ bu¨ ˙er'.

Fig. 46 I-DEAS Machine tool simulation

4.2.4 E-Factory

E-factory permite modelarea digitală a unei întregi întreprinderi, fiind o soluție flexibilă de management a ingineriei tehnologice, care permite o creștere drastică a profitabilității întreprinderii prin îmbunătățirea modului în care produsele sunt fabricate. Sistemul a fost special conceput pentru întreprinderi cu fluxuri tehnologice de producție complexe, întreprinderi al căror succes depinde de aducerea pe piață într-un termen cât mai scurt a unui număr cât mai mare de produse, la un preț redus și de calitate superioară. Avantajele utilizării sistemului E-factory sunt:

- permite echipelor accesul la date cheie în momentul în care aceste date sunt necesare
- avantajează colaborarea eficace și o informare superioară în vederea luărilor de decizii
- permite un răpuns rapid în cazul schimbărilor de design ale produsului
- facilitează distribuirea celor mai bune metode practice (best practices)
- permite afișarea rezultatului impactului deciziilor înaintea repartizării resurselor în vederea producției
- asigură realizarea țelului propus prin aplicarea procesului ales

4.2.5 PLM-Components

UGS´ PLM Components sunt utilitare 'open source' ce permit și ajută la promovarea standardelor industriale și a interoperabilității în cadrul ciclului de viață al produsului. Utilizat de sute de întreprinderi din lumea întreagă, PLM Components permite o desfășure continuă a informațiilor și datelor între diverși parteneri sau liferanți industriali, independent de aplicațiile PLM pe care aceștia le utilizează.

4.2.6 Simulink (MathWorks)

Fig. 47 MathWorks

Dezvoltarea și simularea sistemelor bazate pe axe de timp atât continue cât și discrete. Simulink este o platformă de simulare a sistemelor cu domenii multiple, precum și o dezvoltare bazată pe modele a sistemelor dinamice. Oferind o suprafață interactivă grafică și numeroase blocuri de biblioteci extensibile, permite dezvoltarea, simularea, implementarea și testul sistemelor de automatizare și reglare, prelucrarea semnalelor, sistemelor de comunicații precum și a altor numeroase sisteme dinamice.

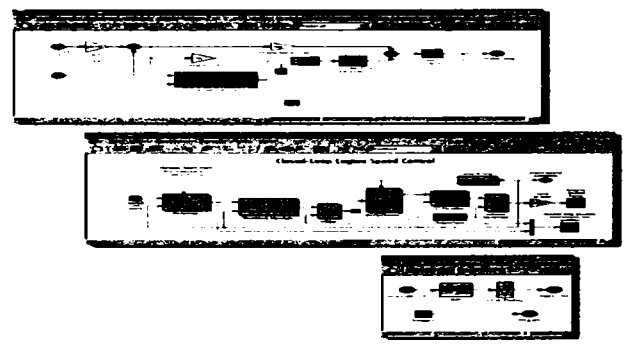


Fig. 48 Simulink

Având în vedere faptul că industria de automobile, cel mai complex produs de larg consum al perioadei actuale, în momentul de față integrează într-un automobil 'high-end' peste 70 de microprocesoare ce controlează motorul, șasiul, siguranța, sistemele de direcție și frânare, precum și cel că în următorii 5 ani numărul acestor elemente o să fie cel puțin dublate, coordonarea acestor elemente, programarea lor și verificarea devine o sarcină tot mai importantă. Inginerii toturor marilor producători de automobile din lume utilizează MATLAB și Simulink în vederea dezvoltării acestor sisteme prin cercetare, analizarea ideilor, modelelarea, simulare și prototipare rapidă a noilor concepții, înaintea trecerii lor în producție. Pentru multe dintre aceste întreprinderi utilizarea acestor utilitare reduce perioada de design la jumătate.

5 Prelucrarea digitală a imaginilor

Unul dintre pașii pricipali ai prototipării rapide este cel al scanării și digitizării. În vederea analizării modului de funcționare a sistemelor de scanare (realizate principial după modul de funcționare al ochiului uman) este necesară o scurtă incursiune în analiza acestui principiu, deoarece acest lucru a dus la elaborarea de algoritmi utilizați la ora actuală în acest domeniu.

5.1 Elemente de Neuropsihologie

5.1.1 Vizualizarea (ochiul uman)

Ochiul uman, deși în principiu relativ simplu, este rezultatul câtorva miliarde de ani de evoluție. Datorită acestui fapt, modul său de funcționare se deosebește de sistemele de vizualizare (ochii) altor viețuitoare ale regnului animal, în special de cel al insectelor. Prin înțelegerea însă a modului său de funcționare au fost puse în evidență punctele slabe din funcționarea acestuia, principiul fiind la ora actuală extins prin utilizarea metodelor în domenii inaccesibile ochiului uman (infraroșu, raze X, etc).

Sistemul optic al ochiului nu este exact centrat. În Fig. 49 este prezentată schematic o secțiune verticală prin ochiul uman. După cum se observă, sistemul optic generează o imagine micșorată și răsturnată a obiectelor din lumea reală. Retina, compusă din celule receptoare sensibile la lumină, care transmit excitațiile exercitate asupra lor sub formă de impuls la ganglioni specializați. Aceștia amplifică semnalul primit și-l transmit prin intermediul nervului optic la creier.

5.1.1.1 Modul de funcționare a retinei

Receptorii existenți sunt de două tipuri, sub forma de bețișoare și de conuri. Bețișoarele, sunt specializate în preluarea excitațiilor luminoase difuze și monocrome (vederea scotopică) iar prin conuri este procesată lumina diurnă, color (vederea fotopică). Se presupune că există trei tipuri de receptori conici, spațial foarte apropiați, fiecare dintre aceștia fiind responsabil de un anumit domeniu spectral al luminii (sensibilitatea spectrală a ochiului uman este limitată la cca. 555nm).

Stratul de receptori ai ochiului uman este constituit din aproximativ 120 milioane de receptori de tip bețișor și 6 milioane de tip con - marea majoritate a acestora din urmă fiind situați în Fovee, zonă pe care aceștia o constituie în mod exclusiv. În această zonă, receptorii din centru au un diametru de circa 2µm, corespunzătoare unui unghi de vedere de aproximativ 0,4'.

Pe de altă parte, imaginea receptată este preprocesată la nivel de retină prin utilizarea unor ganglioni de tip on/off, care reacționează cu semnale ON pentru surse de lumină staționare și respectiv OFF pentru semnale întunecate. Semnalele receptate de retină, acționează zonele centrală și periferică ale acesteia în mod complementar. Astfel, în momentul excitării luminoase a zonei centrale, zona periferică este inhibată, și invers. Astfel, în analiza completă a unei imagini este întotdeauna luat în considerare și mediul înconjurător.

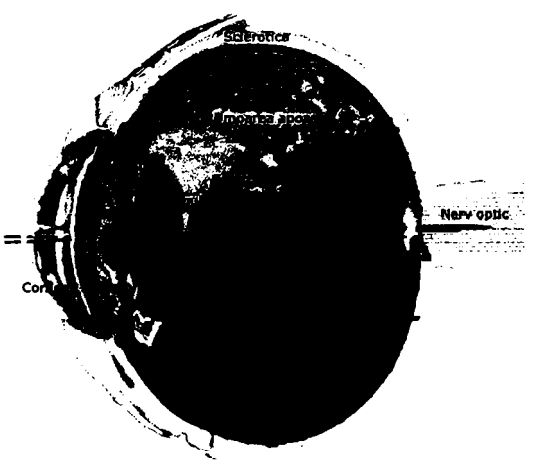


Fig. 49: Ochiul uman

5.1.1.2 Iluzii optice

Dat fiind modul de funcționare al ochiului uman pot fi explicate unele fenomen vizuale și iluzii optice pe care le vom trata în continuare, înțelegerea lor fiind importantă pentru prelucrarea ulterioară a imaginilor, datorită faptului că ochiul uman utilizează aceste mecanisme iar algoritmi utilizați analizează în mod obiectiv imaginile captate electronic, ceea ce în special în perioada inițială și cea de depanare a sistemelor de scanare poate duce la pierderi considerabile de timp.

Luminozitatea

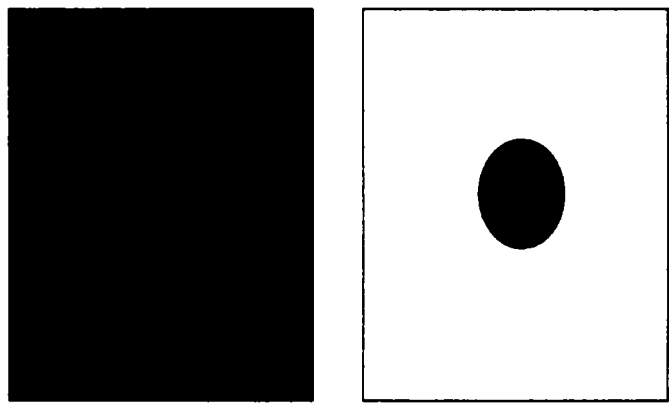


Fig. 50: Iluzie optică - luminozitate

Prin posibilitatea recunoașterii gradului de luminozitate a unei pete de culoare (sau gri), neținând cont de faptul că mediul înconjurător influențează calitatea procesării, avem impresia că cele două pete centrale au grade de luminozitate diferite, ceea ce nu este adevărat.

Culori

La fel ca și în cazul luminozității, o pată de culoare gri pe un fundal verde, pare sa aibă nuanțe de lila.

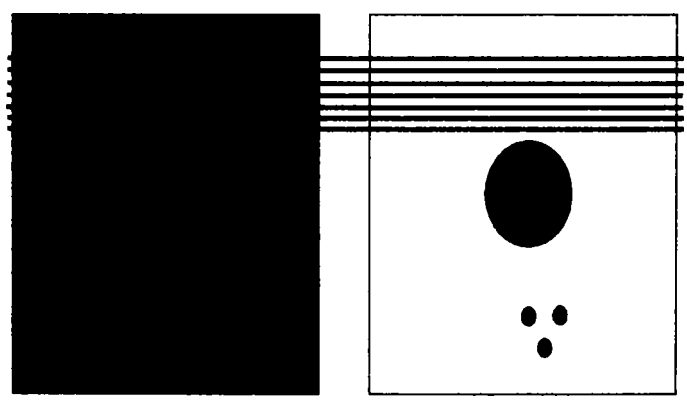


Fig. 51: Iluzie optică - culoare

Forme

Prin analiza imaginilor proiectate, creierul uman reușeste chiar completarea de imagini. Astfel, o persoană

este fără probleme recunoscută cu toate că se află în spatele unei ferestre iar fața îi este acoperită parțial de rama geamului.

Bazat pe acest fapt, apare următoarea iluzie optică, în care, deși în figură sunt reprezentate trei segmente de cerc, vom recunoaște trei cercuri complete, acoperite de un triunghi de culoarea fondului, iar un raster

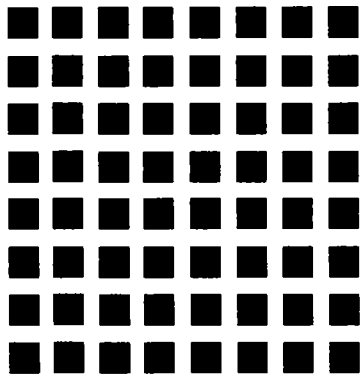


Fig. 52: Iluzie optică - completare

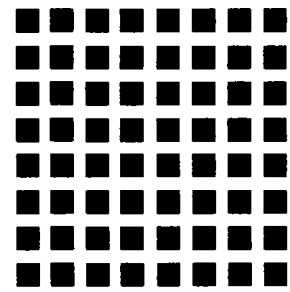


Fig. 53: Iluzie optică - completare 2

constituit din patrate negre pare să conțină în interspații cercuri de culoare gri, care însă dispar în momentul în care încercăm să le focusăm.

Într-un mediu înconjurător constituit din cercuri concentrice, un pătrat pare să aibă laturile curbate iar cercurile din imaginea alăturată par a fi spirale..

Mărimi

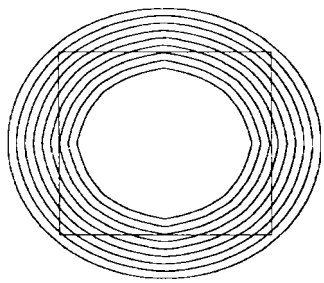


Fig. 55: Iluzie optică - formă

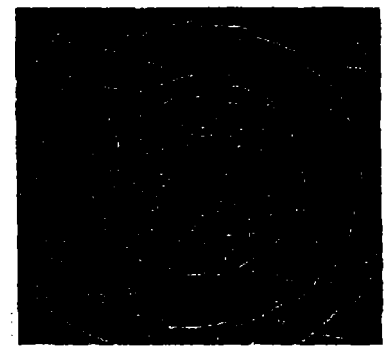


Fig. 54: Iluzie optică - formă 2

Mărimea obiectelor observate este de asemenea influențată de mediul înconjurător, așa cum se poate vedea din Fig. 56. Deși cercurile roșii par a fi de dimensiuni diferite, acest lucru este doar o iluzie optică.

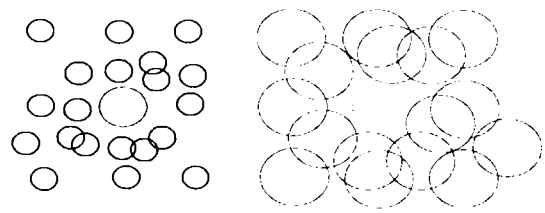


Fig. 56: Iluzie optică - mărime

O altă farsă pe care ne-o joacă vederea, este prezentată în Fig. 57. Cele două segmente roșii sunt în realitate de lungimi egale. În Fig. 59 cele două segmente sunt de asemenea egale ca lungime, însă prin poziționa-

rea săgeților la capete, este obținută o iluzie optică de scurtare a segmentului.

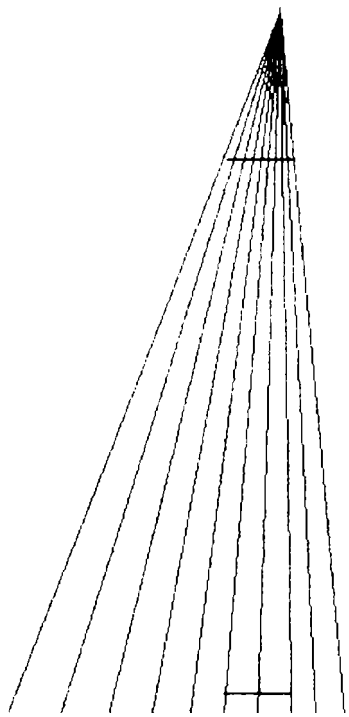


Fig. 57: Iluzie optică - lungime

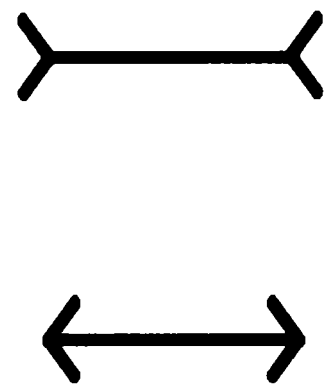


Fig. 58: Iluzie optică - scurtare

Orientări/Direcții

Mediul înconjurător influiențează de asemenea și direcția, respectiv orientarea obiectelor. Deși liniile din Fig. 59 sunt drepte, datorită mediului, par a fi ușor curbate iar cele din Fig. 60 par a fi neparalele.

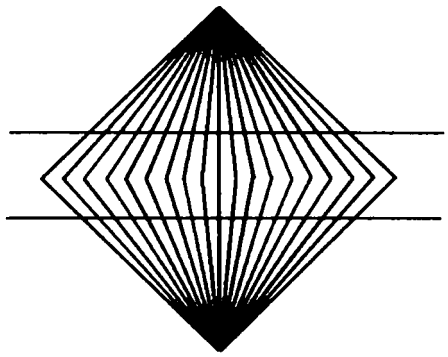


Fig. 59: Iluzie optică - orientare

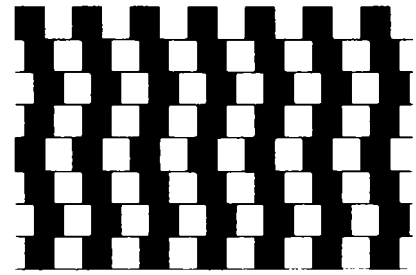


Fig. 60: Iluzie optică - direcție

Figuri imposibile

Percepția vizuală este de asemenea influențată de faptul că în anumite condiții, reprezentări de obiecte ce

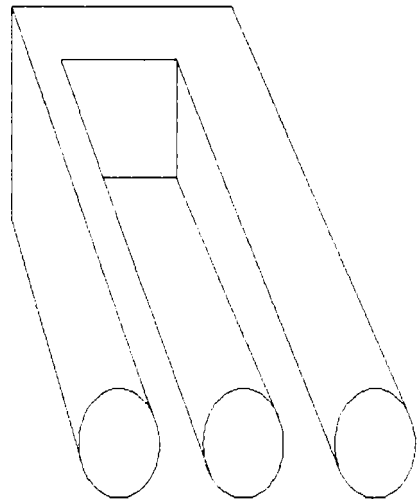


Fig. 61: Iluzie optică -figură imposibilă

par a fi tridimensionale sunt de fapt doar o înlănțuire de linii plane, intenția de a le imagina ca obiecte tridimensionale eșuând. Aceste obiecte nu vor fi obținute prin scanarea obiectelor 3D, dar o scanare plană poate conduce la apariția acestor tipuri de obiecte. Un exemplu binecunoscut este redat în Fig. 61.

O problemă înrudită, rezultată în urma procesului de rendering și datorită setării eronate a parametrilor camerei, este prezentată în figura următoare. Deși modelul este corect, imaginea calculată prin rendering este o figură imposibilă.

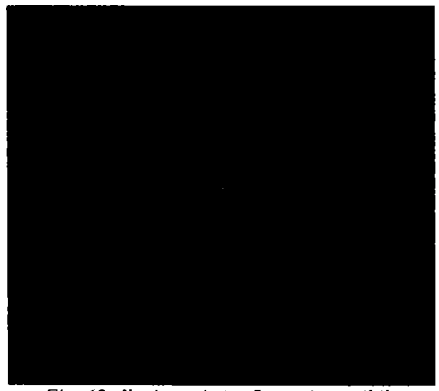


Fig. 62: Iluzie optică – fi_ură imposibilă wireframe

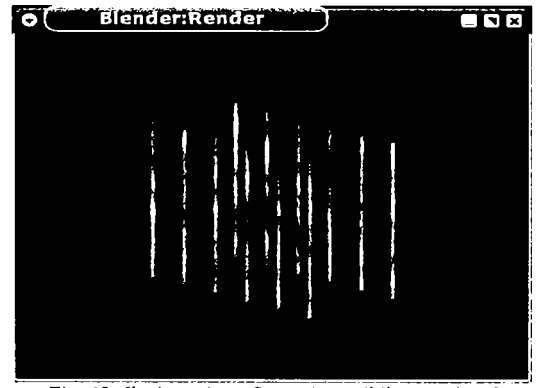


Fig. 63: Iluzie optică - figură imposibilă - rendered

Figuri ambigue

Deşi în principiu nu este vorba de o iluzie optică, dat fiind faptul că printr-o singură analiză a imaginii sunt obţinute două sau mai multe interpretări ale formei reale, acest fenomen trebuie de asemenea amintit. După cum se poate observa din Fig. 64, fără informaţii suplimentare, un program de scanare nu va putea decide dacă este vorba de o concavitate cubulară într-un cub, sau de reprezentarea unui cub situat la intersecţia a trei planuri. Ambele variante sunt posibile.

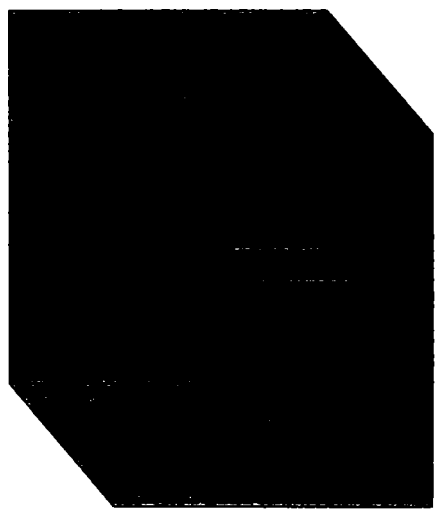


Fig. 64: Iluzie optică - figură ambiguă

5.2 Etapele prelucrării imaginilor digitale

În vederea prelucrării imaginilor, este necesară respectarea următoarelor etape[17]:

- codificarea imaginii
- preprocesarea imaginii
- segmentarea
- extragerea elementelor caracteristice
- clasificarea

5.2.1 Codificarea

Prin codificare se obţine o cuantizare a coordonatelor locale şi a valorilor de gri din imagine. Imaginea va fi reprezentată printr-o matrice ce poate fi prelucrată prin metode numerice, indiferent dacă imaginea este o fotografie digitală, o imagine scanată sau provenită în urma unei radiografii.

În matricea astfel rezultată, vor fi salvate toate informaţiile caracteristice imaginii iniţiale (formă, culori, structură).

5.2.1.1 Noțiuni

Imagini și reprezentări

Imaginea ca funcție

Dat fiind faptul că o imagine nu este altceva decât un obiect 2D a cărui culoare și/sau luminozitate se modifică punct cu punct, putem defini imaginea ca o funcție $b(x, y)$ în variabilele locale amintite și având ca rezultat valoarea de gri, culoarea sau luminozitatea imaginii în punctul respectiv. Aceste valori sunt definite ca fiind numere întregi pozitive, deci:

$$0 \le b(x, y) \le b_{max}, b(x, y) \in \mathbb{N}, b_{max} \in \mathbb{N} \quad (1)$$

Dacă valorile rezultate vor fi salvate într-un byte valoarea maximă este de 255. În cazul Chromatografiei această valoare este de 10 biti (1024 valori de gri). Fiind o reprezentare a lumii reale, valorile fiecărei funcții astfel definite pot avea semificații diferite (viteze, reflexii, absorbții, temperaturi, nivele de înălțime, etc) în funcție de aplicația concretă în care vor fi utilizate.

Din această cauză, în tabela următoare sunt prezentate pe scurt cele mai utilizate valori pentru stocarea informației în imagini digitale, precum și domeniile corespunzătoare.[17][18]

Canale	***Biți/Pixel***	***Valori posibile***	***Domenii de utilizare***
		Imagini în nuanțe de gri	
1	1	[0, 1]	Imagini binare: Documente, Ilustrații, Imprimare
1	8	[0, 255]	Universal: Fotografie, Scanare, Imprimare
1	12	[0, 4095]	Special: Fotografie, Scanare, Imprimare
1	14	[0, 16383]	Profesional: Fotografie, Scanare, Imprimare
1	16	[0, 65535]	Calitate foarte ridicată: Medicină, Astronomie
		Imagini color	
3	24	[0, 255]3	Universal RGB: Fotografie, Scanare, Imprimare
3	36	[0, 4095]3	Special RGB: Fotografie, Scanare, Imprimare
3	42	[0, 16383]3	Profesional RGB: Fotografie, Scanare, Imprimare
4	32	[0, 255]4	CMYK: Imprimante digitale (tipografie)
		Imagini speciale	
1	16	[-32768, 32767]	Imagini positive/negative, prelucrări interne și speciale
1	32	±3.4E38	Medicină, Astronomie
1	64	±1.8E308	Prelucrări interne

Tab. 7: Tipuri și parametri de imagini digitale

Imaginea ca reprezentare discretă

Urmărind prelucrarea pe calculator a imaginilor, vom fi nevoiți să analizăm structura imaginii pixel cu pixel, deci vom rasteriza această imagine dând valori întregi variabilelor funcției amintite anterior. Matricea imaginii, va deveni astfel

$$B(x, y), 0 \le x \le N-1, 0 \le y \le M-1, N \in \mathbb{N}, M \in \mathbb{N} \quad (2)$$

Bineînțeles, pentru reprezentări 3D va fi necesară introducerea și utilizarea unei a treia variabile.

De menționat este de asemenea faptul că, în cazul imaginilor 2D, ordonata este îndreptată în jos - spre deosebire de cea din sistemele geometrice cunoscute - ceea ce implică o eventuală transformare a sistemelor de coordonate în timpul prelucrării imaginilor digitale.

Vecinătăți ale punctelor

Dat fiind faptul că este necesar ca imaginile să fie discretizate în vederea analizării lor, din acestă discretizare rezultă pe de o parte necesitatea rasterizării imaginii, iar pe de altă parte, în funcție de algoritmii utilizați, există posibilitatea alegerii unor tipuri diferite de rastere. Cele mai des folosite sunt de tip rectangular și hexagonal.

Rasterul hexagonal, are avantajul implementării mai ușoare în primul rând ca urmare a algoritmilor existenți mai preciși și mai rapizi, este însă mai rar utilizată datorită faptului că liniile verticale (sau orizontale) – în funcție de direcția de decalare – sunt deranjante pentru ochiul uman având formă de zigzag.

Cea mai simplă vecinătate este cea rectangulară, cu patru vecini direcți, prezentată alături:

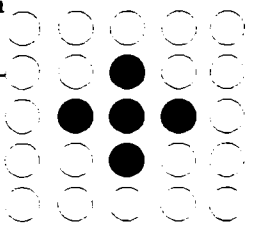


Fig. 65: Vecinătate - 4 puncte

În cazul în care pe lângă aceste patru puncte sunt utilizate și punctele indirect adiacente, punctul va avea 8 vecini (Fig. 66). Problema care apare este însă că distanța dintre punctele luate în considerare nu mai este constantă, între noile puncte aceasta fiind de $a\sqrt{2}$ (considerând distanța dintre puncte ca fiind a). Astfel, algoritmul utilizat este evident mai complex.

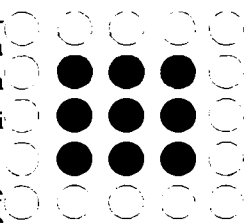


Fig. 66: Vecinătate - 8 puncte

După cum am amintit, cel mai preferabil algoritm este pentru un raster hexagonal cu 6 vecinătăți, caz în care puntele au toate aceeași distanță față de punctul ales, iar calitatea analizei este superioară datorită numărului mai mare de puncte ce intră în calcul (Fig. 67).

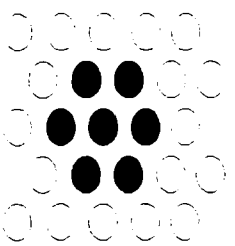


Fig. 67: Vecinătate - 6 puncte

5.2.1.2 Spații de culoare

Monocrom (alb-negru)

În cazul în care imaginea este reprezentabilă doar în alb și negru, spațiul de culoare cel mai potrivit este cel monocrom, căci pentru stocarea informației pixelului este necesar doar un bit. Imaginile binare monocrome sunt foarte bine comprimabile iar algoritmii de lucru asupra lor foarte rapizi.

Gri

În cazul în care imaginea este reprezentabilă prin nuanțe de gri, spațiul acesta de culoare este cel mai recomandabil. În general, acesta este un subspațiu al spațiilor colore care sunt compuse dintr-un număr de trei sau 4 astfel de spații. Cel mai răspândit spațiu de gri este cel cu 256 de nuanțe, dat fiind faptul că un pixel este stocabil într-un byte (8 biți) și astfel viteza de calcul este ridicată.

Color

În funcție de tipul de periferic pentru care sunt utilizate, aceste spații de culoare sunt cele mai complexe utilizabile în vederea prototipării rapide.

După principiul de funcționare, aceste spații pot fi împărțite în

- aditive – în cazul în care, plecând de la culoarea neagră și adăugând culorile de bază în proporții diferite, se obține spectrul complet de culori. Exemplu – RGB care utilizează culorile de bază R(ed), G(reen) și B(lue), deci roșu, verde și albastru. În special, acest spațiu de culoare este utilizat pentru periferice care inițial au culoarea neagră (CRT cu luminoforii stinși) iar prin suprapunerea culorilor de bază de intensități diferite se obțin culorile dorite (valoarea maximă obținută fiind culoarea albă).
- subtractive – în cazul în care culoarea de plecare este cea albă, prin adăugarea de culori de bază de intensități diferite obținându-se celelalte culori. Spațiul de culoare este utilizat în special pentru periferice de tip imprimantă, unde suportul de ieșire (hârtia) are culoarea albă. Prin suprapunerea culorilor de bază la intensitate maximă se obține culoarea neagră.

În continuare ne vom ocupa pe scurt de cele mai răspândite spațiile de culoare și modul de conversie între ele - însă, dat fiind faptul că numărul acestor spații este ridicat, fără pretenția de a fi exhaustivi și în special ne vom referi doar la spațiile de culoare legate de scanare și reprezentare (deci, de exemplu, fără standarde specifice televiziunii).

Standard RGB (sRGB)

RGB este un spațiu de culoare aditiv. Culorile de bază ale acestui spațiu sunt roșu (Red), verde (Green) șu albastru (Blue). Ținând cont de faptul că în sistemele cele mai răspândite valoarea unui pixel este păstrată într-un cuvânt de 32 de biți (pe calculatoarele cu procesoare de 32 de biți având lungimea unei variabile de tip int), reprezentarea internă a spațiului este schematic reprezentabilă în modul următor (unde α este canalul utilizat pentru transparență):

În cazul în care cele trei valori ale culorilor de bază sunt egale, culoarea rezultată este o nuanță de gri, fapt pentru care, aceste tipuri de imagini sunt denumite și imagini color în gri. Singura diferență față de spațiul gri este că nuanțele de gri sunt salvate redundant de trei ori, în fiecare canal al spațiului de culoare.

Fig. 68: Reprezentarea schematică RGB

Spațiul sRGB (standard RGB) a fost conceput cu scopul definirii unui spațiu de culoare care să acopere cât mai precis regulile stabilite în cadrul CIE.

Prin faptul că marea majoritate a monitoarelor și camerelor digitale lucrează în acest sistem, sRGB este pentru scopul lucrării de față spațiul de lucru cel mai important.

Adobe RGB (aRGB)

În principiul identic cu sRGB, doar că în anul 1993 firma Adobe a ajuns la concluzia că, deși pentru monitoare adecvat, spațiul sRGB este pentru imprimare insuficient. Din acest motiv, valoare verde a standardului aRGB este modificată fiind poziționată în

$$x = 0.21 , y = 0.71$$

Informaţii suplimentare pot fi găsite în [19].

CMY şi CMYK

Spaţii de culoare subtractive, utilizate în special pentru imprimarea pe hârtie, culorile de bază fiind Cyan, Magenta şi Yellow (galben).

În practică însă, s-a dovedit de mare importanţă pentru calitatea imprimării utilizarea culorii negre separat, nu doar prin combinarea celor trei culori la saturaţie maximă, în proporţia dată de formula:

$$K = min(C, M, Y)$$

CIEXYZ

În vederea obţinerii unei reprezentări colorimetrice precise, independente de periferiçul pe care are loc reprezentarea, utilizarea unor sisteme calibrate este inevitabilă. Din această cauză şi în acest scop, din anii 1920 CIE (Commision Internationale d'Eclairage) a început elaborarea unui sistem în trei coordonate care în 1931 a fost standardizat. Acest sistem stă practic la baza tuturor spaţiilor de culoare colorimetrice utilizate la ora actuală.

În urma unor măsurători foarte ample şi numeroase făcute sub condiţii stricte, sistemul se bazează pe trei culori de bază imaginare X, Y şi Z, astfel alese încât toate culorile vizibile (cu componente pozitive) să poată fi definite prin acestea, cu toate că cele trei culor primare nu pot fi realizate. Schematic acest spaţiu este reprezentat în figura următoare.

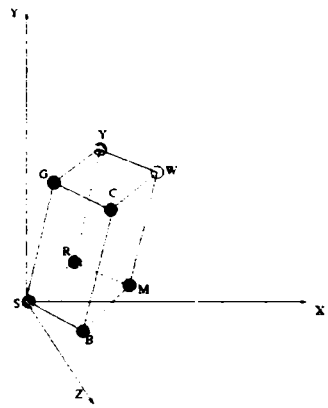


Fig. 69: CIEXYZ

HSV(HSB/HSI) şi HLS

Spaţiul de culoare HSV este definit de componentele Hue (ton de culoare), Saturation (saturaţie) şi Value (luminozitate). Acest spaţiu este adeseori denumit după nomenclatura introdusă de firma Adobe – HSB (Hue, Saturation Brightness) sau rareori cunoscut şi sub denumirea de HSI (Hue Saturation Intensity).

În afara componentelor de bază R, G, şi B, spaţiul utilizează şi cele trei componente ale spaţiului CMY, poziţionând cele şase componente în vârfurile unui hexagon regulat, ca în figura următoare:

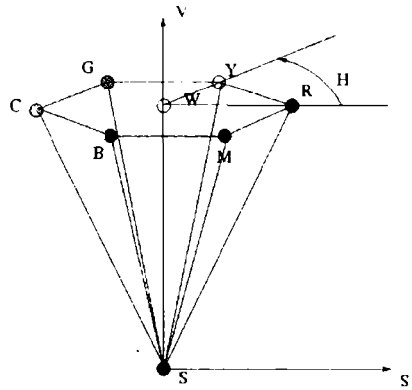


Fig. 70: Spaţiul HSV

HLS (Hue Luminance Saturation) este un spaţiu asemănător, în ceea ce priveşte componenta de Hue chiar identic. Schematic, acest spaţiu poate fi reprezentat după cum urmează:

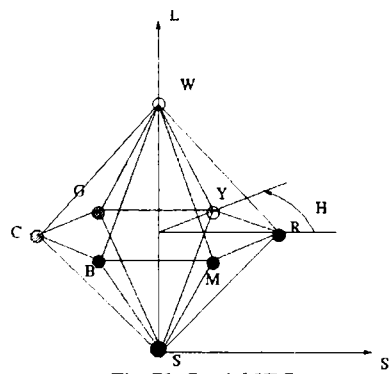


Fig. 71: Spaţiul HLS

Conversii de culoare

În vederea realizării scopului propus, de a prelucra cât mai exact imaginile digitale în vederea îmbunătăţirii performanţelor scanării, este necesară de multe ori conversia datelor prin trecerea de la un spaţiu de culoare la altul, fapt pentru care vom prezenta pe scurt în continuare relaţiile matematice necesare acestor conversii.

Conversia RGB->Gri

În mod normal, din punct de vedere matematic, conversia ar fi exprimabilă prin relaţia:

$$g = \frac{R+G+B}{3}$$

În realitate însă, ochiul uman având sensibilităţi diferite pentru fiecare dintre culorile de bază, relaţia trebuie ponderată prin calcularea unei intensităţi de culoare echivalente, denumită 'Luminanţă', şi care are forma:

$$Y = w_R R + w_G G + w_B B$$

Ponderile de mai sus au valori variabile în funcţie de perifericul utilizat. Astfel pentru televiziunea analogă, aceste valori sunt

$$w_R = 0.229 \, , w_G = 0.587 \, , w_B = 0.114$$

iar pentru codarea digitală a semnalelor valorile:

$$w_R = 0.2125 \, , w_G = 0.7154 \, , w_B = 0.072$$

Aşadar, în realitate, în vederea obţinerii unei imagini gri cu aceeaşi intensitate ca şi cea color, conversia este dată de formula

$$g = Y = [\, w_R w_G w_B \,] \cdot [\, R G B \,]^T$$

Conversia RGB->CMY(K)

Dat fiind faptul că celel două spaţii de culoare se deosebesc prin simplul fapt că unul este aditiv iar celălalt subtractiv, conversia este simplă, şi are forma:

$$C = 1 - R$$
$$M = 1 - G$$
$$Y = 1 - B$$

iar pentru K vom avea:

$$K = min(C, M, Y)$$

Conversia CMY->RGB

La fel de simplă este conversia inversă, deci

$$R = 1 - C$$
$$G = 1 - M$$
$$B = 1 - Y$$

Conversia CMY->CMYK

În cazul că dispunem doar de spaţiul de culoare CMY şi dorim obţinerea corespondentului în CMYK, există trei variante de conversie (ţinând cont de condiţia $K = min(C, M, Y)$ amintită..

Varianta 1:

$$\begin{pmatrix} C' \\ M' \\ Y' \\ K' \end{pmatrix} = \begin{pmatrix} C-K \\ M-K \\ Y-K \\ K \end{pmatrix}$$

Varianta 2:

$$\begin{pmatrix} C' \\ M' \\ Y' \end{pmatrix} = \begin{pmatrix} C-K \\ M-K \\ Y-K \end{pmatrix} \cdot \begin{cases} \dfrac{1}{1-K}, pentru\, K < 1 \\ 1\, altfel \end{cases}$$

$$K' = K$$

Varianta 3:

$$\begin{pmatrix} C' \\ M' \\ Y' \\ K' \end{pmatrix} = \begin{pmatrix} C - f_{UCR}(K) \\ M - f_{UCR}(K) \\ Y - f_{UCR}(K) \\ f_{BG}(K) \end{pmatrix}$$

unde

$$f_{UCR}(K) = s_K \cdot K$$

$$f_{BG}(K) = \begin{cases} 0\, pentru\, K < K_0 \\ K_{max} \cdot \dfrac{K-K_0}{1-K_0}, pentru\, K \geq K_0 \end{cases}$$

Conversia RGB->HSV

Notând cu:

$$C_{hi} = max(R, G, B), C_{lo} = min(R, G, B)$$

vom obţine:

$$S=\begin{cases}\dfrac{C_{hi}-C_{lo}}{C_{hi}} & pentru\ C_{hi}>0\\ 0 & altfel\end{cases}$$

$$V=\frac{C_{hi}}{C_{max}}$$

În cazul în care valorile de R, G şi B sunt egale (gri), vom avea o saturaţie de valoare 0, deci tonul culorii nu este determinat. În celelalte cazuri putem calcula o valoare intermediară a lui H:

$$H'=\begin{cases}\dfrac{G-B}{C_{hi}-C_{lo}} & pentru\ R=C_{hi}\\ \dfrac{B-R}{C_{hi}-C_{lo}}+2 & pentru\ G=C_{hi}\\ \dfrac{R-G}{C_{hi}-C_{lo}}+4 & pentru\ B=C_{hi}\end{cases}$$

Valorile rezultate pentru H' se vor afla în domeniul $[-1,5]$, fapt pentru care formula finală a calculului lui H este următoarea:

$$H=\frac{1}{6}\cdot\begin{cases}(H'+6) & pentru\ H'<0\\ H' & altfel\end{cases}$$

Conversia HSV->RGB

Urmând drumul invers celui prezentat anterior, vom determina mai întâi valoarea H' deci a sectorului în care se află culoarea. Dat fiind faptul că $H\in[0,1]$, vom avea:

$$H'=(6\cdot H)\ mod\ 6$$

Cu ajutorul acestei valori, $0\le H'<6$, vom determina mai întâi unele variabile temporare intermediare, după cum urmează:

$$c_1 = floor(H')$$
$$c_2 = H' - c_1$$
$$x = (1 - S) \cdot V$$
$$y = (1 - S \cdot c_2) \cdot V$$
$$z = (1 - S \cdot (1 - c_2)) \cdot V$$

Cu ajutorul acestor variabile, obţinem valorile normalizate RGB notate cu $R'G'B'$:

$$(R'G'B') = \begin{cases} (V, z, x) & pentru \ c_1 = 0 \\ (y, V, x) & pentru \ c_1 = 1 \\ (x, V, z) & pentru \ c_1 = 2 \\ (x, y, V) & pentru \ c_1 = 3 \\ (z, x, V) & pentru \ c_1 = 4 \\ (V, x, y) & pentru \ c_1 = 5 \end{cases}$$

Scalarea acestor valori în domeniul de definiţie al spaţiului de culoare $[0, N-1]$ (în cazul tipic RGB $N = 256$) devine trivial, prin:

$$R = min(round(N \cdot R'), N - 1)$$
$$G = min(round(N \cdot G'), N - 1)$$
$$B = min(round(N \cdot B'), N - 1)$$

Conversia RGB->HLS

Identic conversiei RGB->HSV, dat fiind faptul că din punctul de vedere al lui H între HLS şi HSV nu există nici o deosebire, putem scrie

$$H_{HLS} = H_{HSV}$$

şi deci putem utiliza formulele definite anterior. Celelalte valori vor fi determinate de

$$L = \frac{C_{hi} + C_{lo}}{2}$$

şi

$$S=\begin{cases} 0 & pentru \quad L=0 \\ \dfrac{C_{hi}-C_{lo}}{2\cdot L} & pentru \quad 0<L\le 0.5 \\ \dfrac{C_{hi}-C_{lo}}{2\cdot(1-L)} & pentru \quad 0.5<L<1 \\ 0 & pentru \quad L=1 \end{cases}$$

Conversia HLS->RGB

Pornind de la considerentul că $H, L, S \in [0, 1]$, în cazul în care L are valori extreme, vom avea:

$$(R', G', B') = \begin{matrix} (0,0,0) & pentru\ L=0 \\ (1,1,1) & pentru\ L=1 \end{matrix}$$

În caz contrar, vom calcula, ca şi în cazurile anterioare, valoarea sectorului de culoare corespunzător, dată de

$$H' = (6 \cdot H) \bmod 6$$

deci $0 \le H' < 6$.

Variabilele de calcul intermediare necesare sunt în acest caz:

$$c_1 = floor\,(H')$$

$$c_2 = H' - c_1$$

$$d = \begin{matrix} S\cdot L & pentru\ L\le 0.5 \\ S\cdot(1-L) & pentru\ L>0.5 \end{matrix}$$

$$w = L + d$$

$$x = L - d$$

$$y = w - (w-x)\cdot c_2$$

$$z = x + (w-x)\cdot c_2$$

Cu aceste notaţii vom obţine valorile normalizate sub forma:

$$(R'G'B') = \begin{cases} (w, z, x) & pentru \ \ c_1 = 0 \\ (y, w, x) & pentru \ \ c_1 = 1 \\ (x, w, z) & pentru \ \ c_1 = 2 \\ (x, y, w) & pentru \ \ c_1 = 3 \\ (z, x, w) & pentru \ \ c_1 = 4 \\ (w, x, y) & pentru \ \ c_1 = 5 \end{cases}$$

Prin scalarea acestor valori în domeniul de definiție al spațiului de culoare $[0,\ N-1]$ (în cazul tipic RGB $N = 256$) vom obține valorile dorite ale spațiului:

$$R = min\,(round\,(N \cdot R'), N-1)$$
$$G = min\,(round\,(N \cdot G'), N-1)$$
$$B = min\,(round\,(N \cdot B'), N-1)$$

5.2.1.3 Tipuri de imagini

Pornind de la spațiile de culoare existente, puten defini în mod corespunzător și tipurile de imagini aferente.

Imagini binare (monocrome)

Imaginile binare, sau alb-negru sunt cele mai utile imagini necesare scanării în vederea prototipării rapide, deoarece algoritmii necesari analizării imaginii și generării modelului sunt simpli și rapizi. Din nefericire însă, aceste imagini sunt de cele mai multe ori rezultatul prelucrărilor anterioare a imaginilor de tip gri sau color. Pentru reprezentarea lor, este necesar un singur bit pe pixel.

Dintre toate tipurile de reprezentare a imaginilor digitale, cea binară este cea mai importantă utilizabilă în vederea analizării formei obiectelor din cadrul imaginii și datorită faptului că pe ele se bazează funcționarea unor filtre, care au capabilitarea de a reacționa selectiv asupra conținutului imaginii de analizat, modificând structura acesteia - motiv pentru care aceste filtre au fost denumit și „Filtre morfologice".

Imagini în nuanțe de gri

În cazul acestor tipuri de imagini, situația este mai complicată chiar decât a imaginilor color, dat fiind faptul că imaginile pot fi salvate în toate tipurile de spații de culoare cunoscute cu excepția celui monocrom, ceea ce face dificilă repartizarea exactă a acestor imagini într-un spațiu de culoare unic determinat.

Din acestă cauză, vom utiliza denumirea de "Imagine în nuanțe de gri" pentru cazul imaginilor în care valoarea pixelilor este cuprinsă între 0 și N_{max}, stocate într-un canal de culoare.

Imagini color

Cele mai răstpândite tipuri de imagini, și de fapt cele care sunt obținute în urma digitizării obiectelor de analizat, dat fiind faptul că majoritatea sistemelor de scanare livrează datele lor de ieșire în acest format. În vederea analizării acestor imagini, este necesară o conversie în tipurile de imagini prezentate anterior.

5.2.2 Preprocesarea

Prin preprocesare se urmăreşte modificarea conţinutului matricii imaginii în vederea operaţiilor ulterioare, însă fără a înlătura elemente caracteristice ale imaginii, ci doar în vederea uşurării aplicării algoritmilor ulteriori, sau în vederea obţinerii unei prelucrabilităţi ulterioare bune.

Dintre principalele scopuri ale preprocesării putem aminti:

- corecturi de iluminare în cazul unei iluminări neomogene
- reducerea perturbaţiilor
- mărirea contrastului
- corectarea deviaţiilor de gri rezultate în urma aplicării efectelor fotografice
- eliminarea erorilor de digitalizare
- normalizarea mărimii, formei şi culorilor
- adoptarea de filtre pentru creşterea nivelului frecvenţelor cu informaţii importante şi reducerea celorlalte

5.2.2.1 Atribute statistice ale imaginilor digitale

După cum se ştie, în vederea obţinerii imaginilor digitale sunt utilizaţi traductori optoelectrici, care ca urmare vor introduce totodată perturbaţii în imagine. În vederea eliminării lor, se recurge la utilizarea analizării stocastice a semnalelor. Astfel din repartiţia nuanţelor de gri se pot extrage caracteristici şi mărimi specifice imaginii de analizat. Ţinând cont de faptul că în majoritatea cazurilor analiza imaginii se face pe un singur canal, valorile conţinute de acesta pot fi considerate ca nuanţe de gri. Fără a reduce generalitatea, în continuare ne vom referi aşadar la nuanţe de gri, chiar dacă de cele mai multe ori canalul analizat aparţine unei anumite culori de bază.

Profilul valorilor de gri

Repartizarea valorilor de gri de-a lungul unei linii, se numeşte profilul valorilor de gri. Din analiza formei acestei linii se pot trage concluzii importante în ceea ce priveşte caracteristicul fundalului, a clarităţii cantelor şi a relaţie dintre informaţie şi perturbaţii. În cazul în care însă este necesară analiza gradului de uniformitate a unei imagini, valoarea medie de gri poate fi de asemenea analizată. Aşadar, în aces tip de analize vom avea două posibilităţi:

- profilul valorilor de gri pe o linie de referinţă – vor fi analizate doar valorile de gri de pe linia de amintită
- profilul cumulat al valorilor de gri – media geometrică a diferenţei dintre suma valorilor de gri aflate deasupra liniei de referinţă şi suma valorilor de gri aflate sub aceasta.

Profilul valorilor cumulate ne va arăta aşadar variaţia valorilor de gri pe verticala imaginii Fig. 72, pe când cel al valorilor de gri variaţia valorilor de-a lungul liniei Fig. 73.

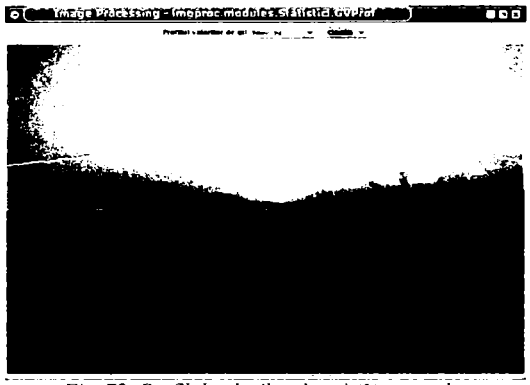


Fig. 72: Profilul valorilor de gri (1) - cumulat

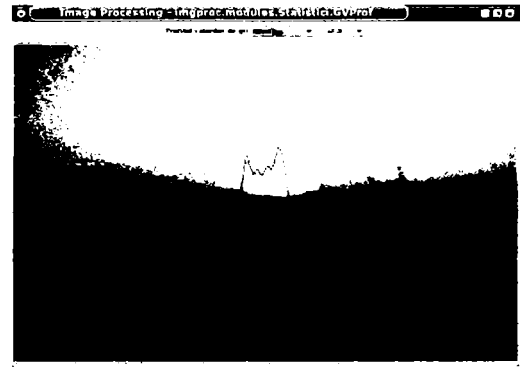


Fig. 73: Profilul valorilor de gri (1) - (A)RGB

În cazul imaginilor la care gradul de separabilitate nu este clar, profilul curbei cumulative va intersecta de mail multe ori axa x (în cazul nostru, mijlocul vertical al imaginii). Un exemplu referitor la acest caz este prezentat în imaginile de mai jos.

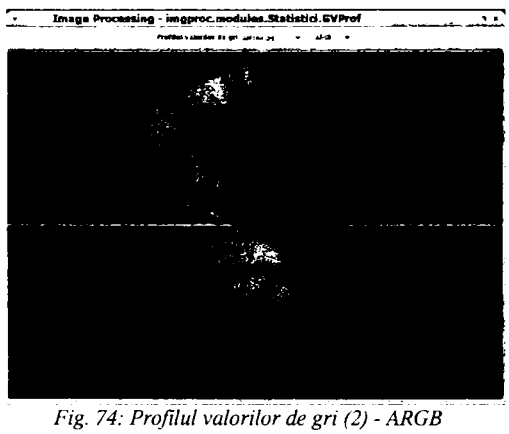


Fig. 74: Profilul valorilor de gri (2) - ARGB

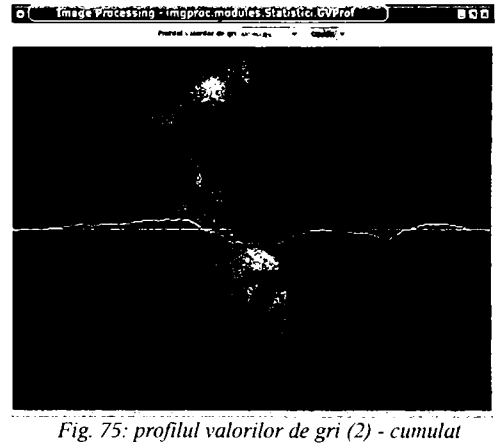


Fig. 75: profilul valorilor de gri (2) - cumulat

De menționat este faptul că în cazul profilului valorilor, axa x a reprezentării este aflată pe ultima linie a imaginii (prima linie de jos în sus), curba având o variație cuprinsă între 0 și 255, iar în cazul profilului cumulativ, ea este la mijlocul imaginii, variația valorilor de gri în acest caz putând avea loc între -255 și 255.

Pentru cazul prototipării rapide, imaginile cele mai importante sunt cele asemănătoare figurilor următoare. Din aceaste figuri se poate observa că variația profilului caracterizează imaginea de prelucrat.

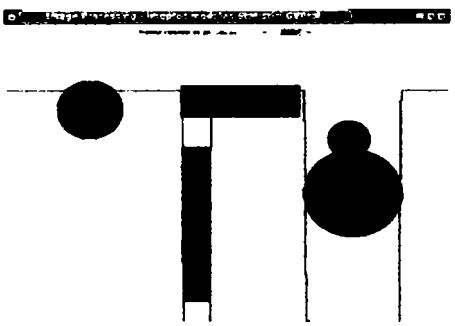


Fig. 77: Profilul valorilor de gri (3)

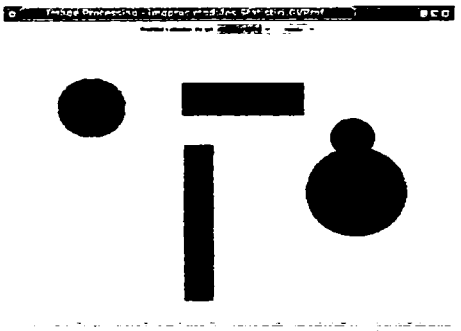


Fig. 76: Profilul valorilor de gri cumulat (3)

Histogramele valorilor de gri

Una dintre cele mai importante mijloace statistice utilizate în prelucrarea digitală a imaginilor este prin utilizarea histogramelor, datorită faptului că prin aceste, imaginea poate fi caracterizată mult mai ușor din punct de vedere al dinamicii, problemelor introduse de mijloacele de captare a imaginii, gradul de distorsiune, precum și eventual îndrumarea în alegerea pașilor următori necesari prelucrării.

Histograma monodimensionala

Luând în considerare o imagine digitală gri cu informația salvată în 8 biți, sau o imagine color în 24 (8R 8G 8B) sau 32 (8A 8R 8G 8B) de biți, pe un canal vom avea posibilitatea de a salva 256 de intensități diferite ale pixelilor. Analizând în continuare pixel cu pixel întreaga imagine, și însumând numarul de intensități diferite întâlnite, vom obține un vector cu 256 de elemente.

La cazul general, Notând acest vector cu H(g) și cu N numărul de pixeli ai imaginii de analizat, vom numi Histograma absolută ca fiind H(g), iar cea relativă h(g) obținută prin:

$$h(g)=\frac{H(g)}{N},\ g\in[g_{min},g_{max}]$$

Această histogramă nu este altceva decât repartiția probabilităților nivelelor de gri și deci vom avea proprietatea:

$$\sum_{g=g_{min}}^{g_{max}} h(g)=1$$

unde g_{min}, g_{max} sunt valoarea minimă și respectiv maximă a intensităților de gri din imagine.

Pe baza acestor informații se pot extrage valori statistice relevante (momente centrale de ordinul n), care caracterizează imaginea de analizat.

Forma generala a momentelor centrale de ordinul n este dată de relația:

$$M_n=\sum_{g=g_{min}}^{g_{max}} (g-\bar{g})^n h(g)$$

Aceste momente, precum și unele valori derivate ale acestora sunt:

n = 1 - Media aritmetică

$$\bar{g} = \sum_{g=g_{min}}^{g_{max}} g\,h(g)$$

n = 2 – Varianța sau abaterea media pătratică (σ^2)

$$M_2 = \sigma^2 = \sum_{g=g_{min}}^{g_{max}} (g-\bar{g})^2 h(g)$$

n = 3 – Înclinarea (skewness) gradul de asimetrie

$$M_3 = \sum_{g=g_{min}}^{g_{max}} (g-\bar{g})^3 h(g)$$

Prin acesta, este constatată abaterea de la forma normală Gauss a histogramei în sensul deplasării spre stânga sau dreapta. Acest moment se poate scrie și sub o formă adimensională:

$$M_3' = \frac{M_3}{\sqrt{M_2}^3}$$

O distribuție la stânga are o valoare pozitivă a înclinației, iar o deplasarea la dreapta, o valoare negativă.

n = 4 – Excesul (kurtosis)

$$\frac{M_4}{M_2^2} = 3$$

Asemeni inclinării, excesul ne arată deplasarea imaginii față de repartiția normală Gauss, de dat aceasta însă în sus sau în jos. O măsură adimensională a acestui moment, este dată de:

$$M_4' = \frac{M_4}{M_2^2} - 3$$

Valoare 3 este extrasă, datorită faptului că în cazul unei repartiții normale avem $\frac{M_4}{M_2^2} = 3$ iar astfel pentru forma normală vom avea valoarea de 0.

În figurile următoare sunt prezentate câteva exemple de histograme monodimensionale în nivele de gri și RGB.

În mod identic se poate obține histograma valorilor de gri pentru fiecare canal al imaginii (în cazul prezentat în pentru RGB).

Fig. 78: Histogramam valorilor de gri cumulată

Fig. 79: Histogramam valorilor de gri

Histograma Bidimensională

În mod asemănător, prin separarea canalelor de culoare, se pot obține histograme 2D ale imaginii de anali-

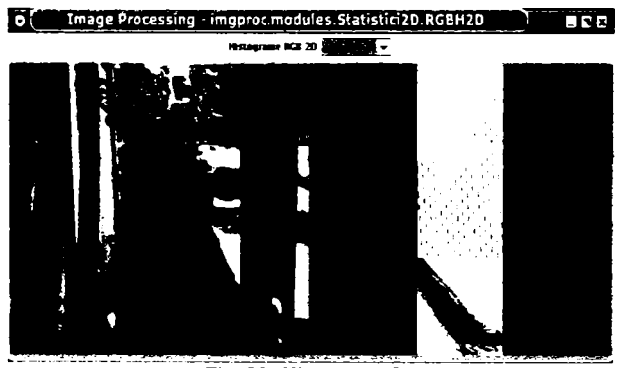


Fig. 80: Histograme 2D

zat. În partea stângă a Fig. 80 este prezentată histograma 3D a imaginii.

Matricea de transfer a valorilor de gri (Coocurence)

Unul dintre dezavantajele cele mai importante ale histogramelor de gri, este dat de faptul că pentru imagini diferite se poate obține aceeași histogramă de gri în cazul în care numărul valorilor de gri este la fel repartizat pe imaginile de analizat, neținându-se cont de conținutul imaginii (forma). Un exemplu foarte simplificat este prezentat în cele trei figuri care urmează:

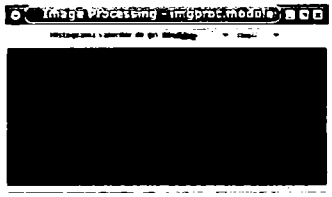


Fig. 81: Coocurență 1

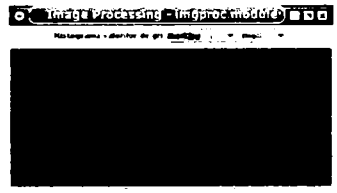


Fig. 82: Coocurență 2

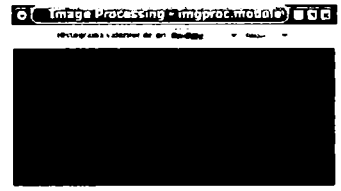


Fig. 83: Coocurență 3

După cum se poate observa, deși imaginile sunt complet diferite (variația poziției zonei gri), histograma de gri este identică.

Din această cauză, în vederea luării în considerare și a structurii imaginii, este utilizată matricea de coocurență, numită și matricea de transfer a valorilor de gri. Pentru aceasta, se va lua în considerare și relația între două puncte ale imaginii aflate la o distanță $\delta(\Delta x, \Delta y)$ impusă.

Notând matricea de coocurență cu $C_\delta(i,j)$ compusă din probabilitățile ca două puncte $I_1(x,y)$ și $I_2(x+\Delta x, y+\Delta y)$ să posede valoarea de gri i și respectiv j. În matricea de coocurență se vor afla deci probabilitățile absolute a perechilor de puncte cu valorile de gri amintite mai sus, aflate la o distanță mai mică sau egală cu δ. De menționat este de asemenea faptul că atât lungimea cât și orientarea lui δ vor determina matrici, fapt pentru care este recomandabil ca valorile parametrilor Δx și Δy să fie reduse. De asemenea, dimensiunea matricii ce coocurență este dependentă de numărul valorilor de gri, fapt pentru care alegerea unor trepte de valori este de asemenea indicată. În practică, se utilizează în general valori între 8 și 64 de nivele de gri – variabilă notată în continuare cu Q. După [20], din matricea de coocurență se vor obține următoarele caracteristici:

Energie (Momentul unghiular secundar – angular secundar moment - ASM)

$$T_1(\delta) = \sum_{i=1}^{Q} \sum_{j=1}^{Q} C_\delta(i,j)^2$$

Contrast

$$T_2(\delta) = \sum_{i=1}^{Q} \sum_{j=1}^{Q} (i-j)^2 C_\delta(i,j)$$

Prin acest atribut, este caracterizată inerția matricii de-a lungul diagonalei principale.

Entropie

Atribut identic cu mărimea entropiei din teoria informației, și este proporțional cu gradul de omogenitate al imaginii.

$$T_3(\delta) = -\sum_{i=1}^{Q} \sum_{j=1}^{Q} C_\delta(i,j) \log C_\delta(i,j)$$

Corelație

$$T_4(\delta)=\frac{1}{\sigma_m\sigma_n}\left(\sum_{i=1}^{Q}\sum_{j=1}^{Q} ijC_\delta(i,j)-\mu_m\mu_n\right)$$

unde μ_m, μ_n sunt valorile medii, iar σ_m, σ_n abaterile medii pătratice ale p_m, și p_n.

$$p_m(i)=\sum_{j=1}^{Q} c_\delta(i,j)$$

și

$$p_n(j)=\sum_{i=1}^{Q} c_\delta(i,j)$$

Corelația este proporțională cu contrastul imaginii.

5.2.2.2 Conținutul informativ al imaginilor digitale

Teoria informației caracterizează acest subiect în trei tipuri diferite:

- sintactice
- semantice
- pragmatice

Informațiile sintactice depind de realitarea fizică și sunt singurele care, fiind măsurabile, pot fi utilizate în analizarea și prelucrarea imaginilor digitale. Informațiile semantice descriu conținutul informație și sunt dependente de experiența observatorului, sunt deci subiective. Cu atât mai puțin pot fi utilizate informațiile pragmatice, care descriu rezultatul influenței informației asupra observatorului.

Idea principală a analizei conținutului informativ al imaginilor digitale rezidă din faptul că în interiorul imaginii, nu toate informațiile sunt relevante, sau interesante și este de aceea necesară o analiză atât a cantității de informație existentă în imagine, precum și a modului de prelucrabilitate a acesteia.

Știind că o sursă de informație emite n mesaje, acestea ne vor parveni sub forma unui șir exprimabil prin $(m_1\, m_2 \cdots m_n)$. Probabilitatea de receptare a mesajului m_i este așadar

$$p_i=\frac{n_i}{n}$$

iar suma tuturor evenimentelor rezultate în urmă interceptării tuturor semnalelor este bineânțeles dată de evenimentul sigur, reprezentând probabilitatea ca un semnal să fie interceptat în momentul în care toate mesajele au fost interceptate. Această constatare poate fi scrisă sub forma:

$$\sum_{i=1}^{n} p_i=1$$

Pe de altă parte însă, cu cât probabilitatea apariției mesajului m_i este mai mare, cu atât este mai mic conținutul informativ al mesajului. Astfel, în cazul unei imagini digitale, dacă secvența de gri a unor pixeli alăturați este constantă (255 255 255 255), informația devine tot mai neinteresantă, fenomen numit și redundanță. Așadar, odată cu creșterea redundanței vom avea o scădere a conținutului informativ, și invers.

Cantitatea de informație este definită în teoria informației ca fiind dată de formula:

$$I = k \log \frac{1}{p_i} = -k \log p_i$$

unde prin alegerea adecvată a lui k în funcție de sistemul informativ în care are loc analiza, este definită unitatea cantității de informație.

În cazul unui sistem binar, deci alegând logaritmul în baza 2 și $k = 1$ vom obține unitatea de cantitate de informație a unui bit:

$$I = \log_2 \frac{1}{p_i} = -\log_2 p_i$$

Conținutul informativ

Utilizând histograma valorilo de gri prezentată anterior, putem cu ușurință determina conținutul mediu informativ al unui bit al unei imagini digitale, acesta fiind dat de formula:

$$I_{med} = \sum_{g=1}^{N \times M} h(g) \cdot I(h(g)) = -\sum_{g=1}^{N \times M} h(g) \cdot \log_2 (h(g))$$

unde $h(g)$ este valoarea corespunzătoare a nivelului g din histograma relativă a valorilor de gri.

Conținutul informativ al imaginii următoare este de 7.812, deci utilizând 253 din 256 nuanțe de gri, imaginea utilizează într-un mod rațional plaja de valori oferită de o codificare de 8 biți pe pixel.

Fig. 84: Conținutul informativ al imaginii

Diferența pixelilor învecinați

De cele mai multe ori, pixelii învecinați din cadrul unei imagini digitale au valori egale sau apropiate, fapt care poate fi reprezentat prin imaginea diferenței acestor pixeli, calculată prin formula:

$$I'(u,v) = I(u,v) - I(u-1,v) + 128$$

Valoarea de 128 a fost adăugată ținând cont de faptul că plaja de valori a nuanțelor de gri a unei imagini codate pe 8 biti este 256 și astfel ne vom poziționa în imaginea rezultată în zona centrală a domeniului de valori.

După cum se poate observa din Fig. 85, imaginea diferenței valorilor de gri necesită doar patru din cei 8 biți pentru codificare.

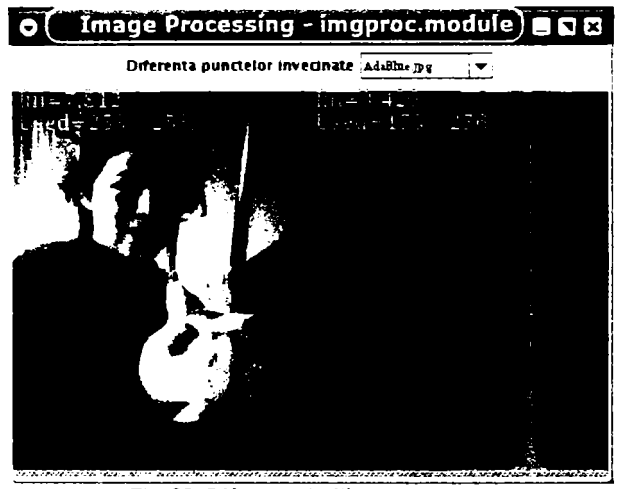


Fig. 85: Diferența pixelilor învecinați (1)

În cazul imaginilor cu fluctuație mai redusă a valorilor pixelilor învecinați, conținutul informativ este și mai redus, după cum reiese din figura următoare, unde conținutul informativ este de numai trei biți pe pixel din cei 8 posibili.

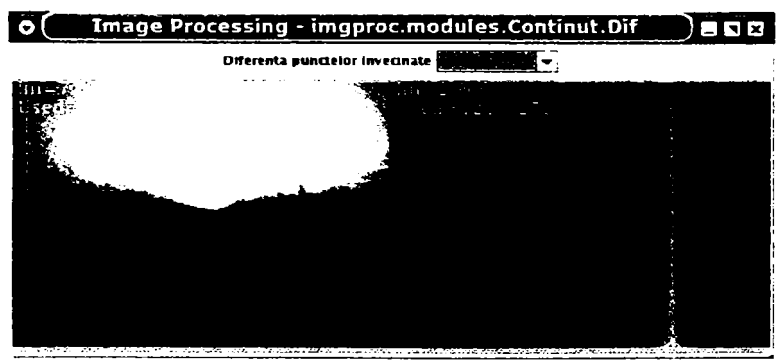


Fig. 86: Diferența pixelilor învecinați (2)

Nivele de biți

Datorită analizelor făcute anterior, am constatat că nu toți biții unui pixel al imaginii conțin informații uti-

le. În exemplul din figura ce urmează sunt prezentate separat nivelele de biți ale fiecărui pixel al imaginii (în cazul de față 8).

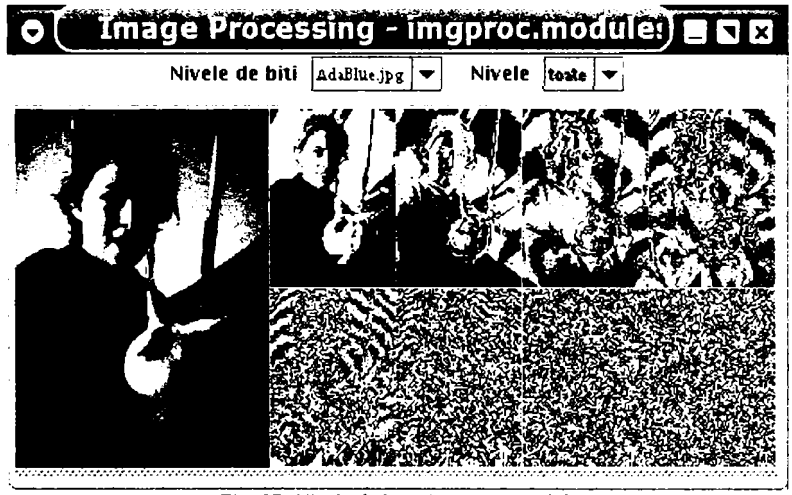


Fig. 87: Nivele de biți (1) - toate nivelele

Prin suprapunerea a doar patru nivele de biți, după cum reiese din analiza conținutului informativ al imaginii prezentate anterior, imagine rezultată va arăta în modul următor:

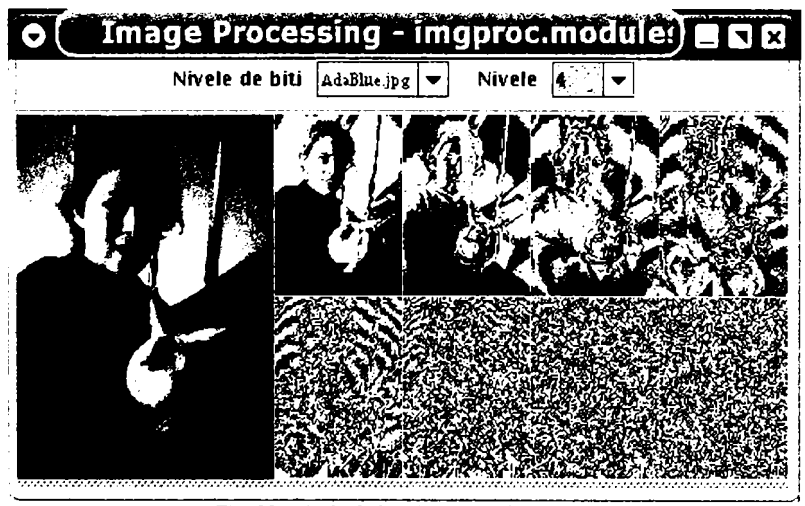


Fig. 88: Nivele de biți (1) - primele 4 nivele

Deși observabile, diferențele dintre Fig. 87 și 88 sunt relativ reduse, iar imaginea rezultată este suficient de clară pentru a-i putea fi înțeles 'mesajul'.

Prin suprapunerea a doar trei nivele, imaginea rezultată va fi nesatisfăcătoare, iar prin suprapunerea a cinci nivele vom avea numeroase informații redundante.

Comparativ, este prezentat în continuare (Fig. 90) cazul celeilalte imagini analizate anterior.

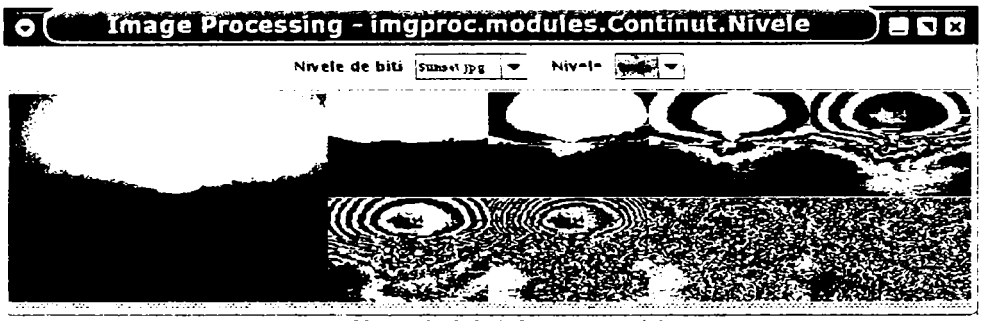


Fig. 89: Nivele de biți (2) - toate nivelele

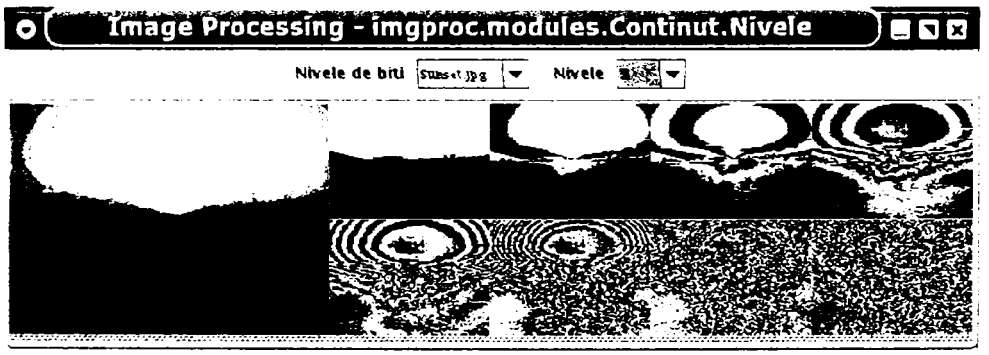


Fig. 90: Nivele debiți (2) - trei nivele

5.2.2.3 Operații punctuale asupra imaginilor

Prin operații punctuale se înțeleg operațiile asupra imaginilor, care modifică doar valorile elementelor imaginii și nu dimensiunea, geometria sau structura acestora. Astfel, valoarea fiecărui punct $I'(u,v)$ nou obținut, va fi dependent de valoare punctului inițial $I(u,v)$ al imaginii. Principial, aceste operații pot fi împărțite în două tipuri:

- Operații omogene – funcția este independentă de poziția pixelului în imagine $I'(u,v)=f(I(u,v))$
- Operații inomogene – funcția este dependentă de poziția punctului în imagine (e.g. modificarea selectivă a contrastului sau luminozității în vederea îmbunătățirii calității imaginii prin remedierea erorilor de scanare) $I'(u,v)=f(I(u,v),u,v)$

Intensitatea imaginii - Contrast și luminozitate

Cele mai simple operații punctuale sunt cele de modificare a intensității imaginii. Funcțiile omogene corespunzătoare pot fi definite în modul următor:

$$I'(u,v)=I(u,v)*C$$

pentru contrast, unde C este o constantă a contrastului - o creștere cu 25% a contrastului este obținută pentru $C=1.25$, iar o diminuare pentru $C=0.75$.

Asemănător, se poate defini funcţia de luminozitate prin:

$$I'(u,v)=I(u,v)+L$$

unde L este constanta de luminozitate.

În amblele cazuri (ca de fapt pentru orice operaţii efectuate asupra imaginilor), este esenţială validarea valorilor obţinute, pentru a nu depăşi limita maximă şi respectiv minimă a valorilor admise (2^k-1, unde k este valoarea în biţi a reprezentării interne a unui canal). Această validare, denumită şi 'clamping', are pentru cazul unui canal de 8 biţi forma:

```
if(255 < p) then p=255;
if(0 > p) then p=0;
```

De foarte multe ori, este însă necesară o modificare automată a contrastului sau luminozităţii, în vederea expandării domeniului de valori ale imaginii pe toată gama pusă la dispoziţie de spaţiul de culoare. Considerând minimul şi cea maximul valorilor de gri ale unei imagini ca fiind notate cu p_{min} şi p_{max} precum şi cele admise de spaţiul de culoare s_{min} şi s_{max}, funcţia de autocontrast este definită de formula:

$$I'(u,v)=\left(I(u,v)-p_{min}\right)\frac{s_{max}-s_{min}}{p_{max}-p_{min}}$$

sau, dacă utilizăm cazul amintit anterior, unde $s_{min}=0$ iar valoarea maximă este $s_{max}=255$

$$I'(u,v)=\left(I(u,v)-p_{min}\right)\frac{255}{p_{max}-p_{min}}$$

În practică însă, prin utilizarea simplă a acestor formule, se poate ajunge la valori extreme ale anumitor puncte, fapt pentru care se recurge la utilizarea unui prag de saturaţie la ambele capete ale scării valorilor spaţiului de culoare. Valoarea acestui prag este cuprinsă între 0.5% şi 1.5%. În aces caz, funcţia este definită prin:

$$I'(u,v)=\begin{cases} s_{min} & pentru\ I(u,v)\le s_{lo} \\ (I(u,v)-s_{lo})\dfrac{s_{max}-s_{min}}{s_{hi}-s_{lo}} & pentru\ s_{lo}\le I(u,v)\le s_{hi} \\ s_{max} & pentru\ I(u,v)\ge s_{hi} \end{cases}$$

Inversarea

Inversarea este o operaţie punctuală prin care valoarea pixelului iniţial este negată, iar pe de altă parte, pentru a rămâne în domeniul de valori, o valoare constantă este adăugată. Formula de calcul este dată de:

$$I'(u,v)=-I(u,v)+p_{max}=p_{max}-I(u,v)$$

unde în cazul amintit mai sus (8 biţi pe canal pixel) $p_{max}=255.$

Thresholding

Prin această operaţie se urmăreşte transformarea în trepte a imaginii. Prin alegerea unei valori de prag alese, toate valorile pixelilor mai mari decât această valoare vor fi setate la o valoare t_1 aleasă a de spaţiului de culoare, iar cele mai mici vor fi setate pe valoarea t_0. Funcţia corespunzătoare este deci dată de:

$$I'(u,v)=\begin{cases} t_0 & pentru\ I(u,v)<s_{th} \\ t_1 & pentru\ I(u,v)\geq s_{th} \end{cases}$$

O aplicaţie foarte des utilizată, este binarizarea imaginii, caz în care valorile pentru $t_0=0$ şi $t_1=255$.

Corectura Gamma

În cadrul presupunerilor pe care le-am făcut până acum, ne-am referit la mărimile utilizate (culoare, luminozitate, contrast) ca fiind mărimi liniare. În realitate însă, raportul dintre mărimile reale şi cele utilizate nu este liniar, dat fiind faptul că sensibilitatea ochiului uman nu este aceeaşi pentru toate culorile, şi după cum am observat în partea de iluzii optice, cantitatea de culoare este dependentă de mediul înconjurător. În cazul imprimării imaginii, raportul dintre valoarea pixelului şi cantitatea de toner necesară este de asemenea neliniar. Noţiunea de Gamma provine iniţial din domeniul fotografie 'analoage' unde între gradul de expunere al negativului şi gradul de întunecime al imaginii există o dependenţă aproximativ logaritmică, dar pe o foarte mare porţiune curba fiind aproximativ liniară şi monoton crescătoare. Tangenta unghiului de creştere a fost în mod tradiţional denumită ca valoarea Gamma a filmului.

Aceeaşi problemă a apărut şi în cadrul televizoarelor şi monitoarelor datorită neliniarităţii determinate de tuburile catodice, fapt pentru care şi în acest domeniu noţiunea de Gamma a fost adoptată în vederea eliminării efectelor nedorite astfel apărute.

Funcţia Gamma este definită prin formula:

$$f_\gamma(x)=x^\gamma,\ \forall\ x\in\mathbf{R},\gamma>0$$

şi are în domeniul [0..1] reprezentarea pentru valorile 0.05, 0.2, 0.5, 1.0, 2.0, 5.0 şi 20.0 următoarea reprezentare grafică:

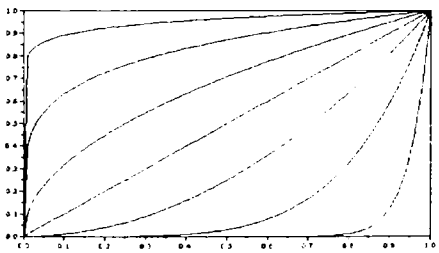


Fig. 91: Funcţia Gamma

Valorile concrete necesare pentru aparate, sunt în general date de manufactor şi sunt determinate experimental.

Din analiza funcție Gamma, rezultă modul de funcționare a corecturii, mod care schematic este prezentat în Fig. 93:

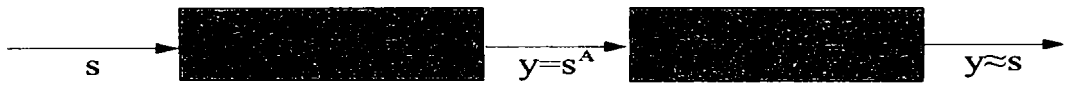


Fig. 92: Corecția Gamma

Notând astfel cu s semnalul de intrare și cu y cel de ieșire, vom avea teoretic:

$$y=(s^{A})^{\frac{1}{A}}=s$$

Datorită comportamentului funcției în apropierea originii apar în realitate probleme de compensare, analizabile prin intermediul primei derivate a funcției Gamma, dată de:

$$f'_{\gamma}(x)=\gamma\, x^{(\gamma-1)}$$

în care deci vom avea

$$f_{\gamma}'(0)=\begin{cases}0, & pentru\ \gamma>1\\ 1, & pentru\ \gamma=1\\ \infty, & pentru\ \gamma<1\end{cases}$$

Din acest motiv, este utilizată în apropierea valorii de 0, până la o valoare x_0, o corectură liniară, funcția Gamma arătând în cele din urmă ca în formula următoare:

$$f_{\gamma,x_0}(x)=\begin{cases}\dfrac{\gamma}{x_0(\gamma-1)+x_0^{1-\gamma}}x, & pentru\ 0\le x\le x_0\\ (\dfrac{1}{x_0^{\gamma}(\gamma-1)+1})x^{\gamma}-\dfrac{1}{x_0^{\gamma}(\gamma-1)+1}+1, & pentru\ x_0<x\le 1\end{cases}$$

Esențial pentru alegerea lui x_0 este ca atât valorile funcțiilor componente cât și a derivatelor acestora în acest punct să fie egale.

Liniarizarea histogramei

Prin liniarizarea histogramei se urmărește obținerea unei repartiții cât mai egale a nivelelor de gri în cadrul imaginii. În principiu, histograma rezultantă va tinde să aibă o formă dreptunghiulară, iar cea cumulată - triunghiulară. Această operație este utilă, de exemplu în cazul necesității comparării a două imagini cu distribuții de intensități diferite. Datorită condițiilor concrete (existența vârfurilor în histogramă) o aplatizare

ideală a imaginii nu este posibilă, repartiția valorilor de gri apropiindu-se însă de linia ideală.

Problema care se pune, este deci de a găsi funcția corespunzătoare operației de liniarizare. Formula de calcul prezentată în continuare este demonstrată în [21]:

$$f_{eq}(p) = floor\left(\overline{H}(p)\frac{K-1}{MN}\right)$$

unde:

K – Numărul de biți necesar reprezentării canalului spațiului de culoare

M – numărul de linii ale imaginii

N – numărul de coloane ale imaginii

Transformarea culorilor

Prin transformarea culorilor, se modifică separat canalele spațiului de culoare.

Cuantizarea valorilor de gri

După cum am văzut anterior, în vederea obținerii unei valori rezonabile a matricilor de calcul, este necesară împărțirea valorilor de gri în domenii discrete. În general, pentru valori mici de cuantizare, modificările obținute sunt imperceptibile pentru ochiul uman, având însă o importanță deosebită în memorarea imaginilor pe suporți magnetici, permițând o comprimare calitativ superioară.

Din figurile următoare, se poate cu ușurință observa că pentru o valoare de 8 (deci doar 32 de valori de gri din 256 – în imaginea de mai jos pe canal, deoarece imaginea este în nuanțe de gri, dar spațiul de culori utilizat este totuși RGB), percepția ochiului uman este puternic influențată de pixelii înconjurători.

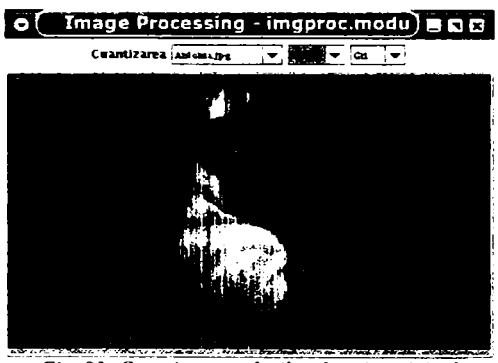


Fig. 93: Cuantizarea valorilor de gri - original

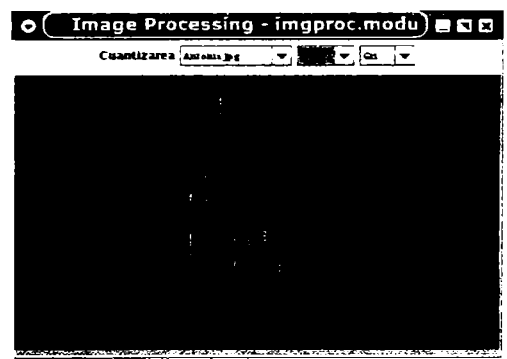


Fig. 94: Cuantizarea valorilor de gri - 8

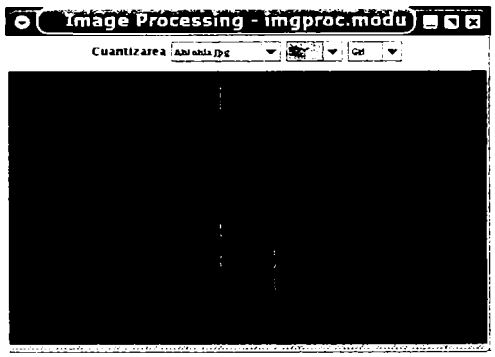


Fig. 95: Cuantizarea valorilor de gri - 32

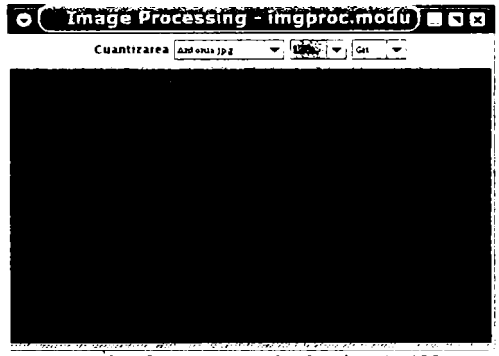


Fig. 96: Cuantizarea valorilor de gri - 128

Tabele de lookup (Lookup tables - LuT)

Datorită faptului că operațiile punctuale necesită de multe ori operații complexe asupra valorilor pixelilor, precum și a numărului ridicat de pixeli din cadrul imaginii, în practică se recurge la utilizarea tabelelor de lookup. Aceste tabele, sunt o transformare discretă $K \rightarrow K$ între două valori ale pixelilor, sau altfel formulat:

$$L:[0,\ K-1] \rightarrow [0,\ K-1]$$

Astfel, definind funcții definite și cu valori în domeniile de mai sus, operațiile necesare sunt efectuate o singură dată, transformările fiind reduse la o căutare a valorii în tabelele de lookup, așadar

$$I'(u,v) \leftarrow L[I(u,v)]$$

Din acest motiv, majoritatea filtrelor utilizate se bazează pe utilizarea tabelelor de lookup.

Alpha blending

Alpha blending este o metoda simplă de amestecare a conținutului mai multor - în principal două - imagini, una de fond (background - BG) și una de front (foreground - FG) prin utilizarea unei valori de transparență α. Formula generală este dată de:

$$I'(u,v) = \alpha I_{BG}(u,v) + (1-\alpha) I_{FG}(u,v), \quad 0 \le \alpha \le 1$$

În cazul în care $\alpha = 0$ imaginea de front este opacă iar pentru $\alpha = 1$ ea este transparentă, permițând astfel observarea pixelilor imaginii de fond.

5.2.2.4 Filtre

Rasterizarea rezoluției locale

Prin cuantizarea rezoluției locale, se urmărește reducerea numărului de pixeli cu valori de gri diferite. În general, pentru a nu deforma imaginea, rasterele sunt pătrate, având valori în puteri ale lui 2 (4x4, 8x8, 16x16...) Exemple corespunzătoare sunt prezentate în continuare:

Fig. 97: Rasterizare 2x2

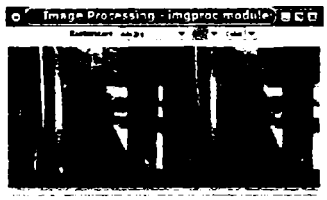


Fig. 98: Rasterizare 4x4

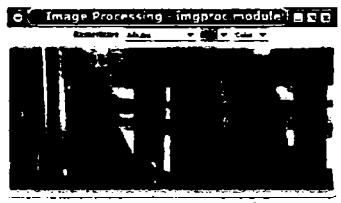


Fig. 99: Rasterizare 8x8

Rasterizarea este pe de altă parte o caracteristică de importanță majoră a aparatelor de digitizare, dat fiind faptul că ea determină rezoluția maximă de scanare a acestora.

Deși în principiu la fel construit ca și celelalte filtre, datorită faptului că algoritmul de lucru nu este bazat pe LuT, a fost în cadrul acestei lucrări separat tratat de celelalte filtre.

Filtre liniare

Deși în practică sunt utilizate în principal filtre de dimensiune 3x3 sau 5x5, aceste dimensiuni nu sunt limitate, iar variația valorilor lor este nelimitată. În general, există sisteme care lucrează cu valori ale matricii între 0 și 1, iar altele doar în valori întregi, diferența dintre acestea este irelevantă, cele două sisteme fiind echivalente prin utilizarea unei constante de scalare.

Filtrul "Box"

Unul dintre cele mai simple filtre liniare, dar în general neadecvat în vederea netezirii imaginilor, datorită faptului că pixelii învecinați centrului au aceeași valoare cu acesta, iar marginile au căderi abrupte. Schematic, filtrul poate fi reprezantat ca în figura următoare:

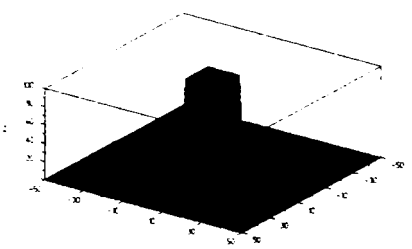


Fig. 100: Filtre liniare - Box

Așadar, reprezentarea matricială este pentru cazul 5x5:

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Tab. 8: Filtre liniare – Box

Un exemplu de filtrare de acest tip cu matrici de tip 3×3 și 5×5 este prezentat în continuare:

Fig. 101: Filtrul "Box" 3x3

Fig. 102: Filtrul "Box" 5x5

Filtrul "Gauss"

Matricea de filtrare a unui filtru de acest tip este o discretizare a funcției Gauss

$$G_\sigma(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

unde abaterea medie pătratică σ este raza elementului funcției.

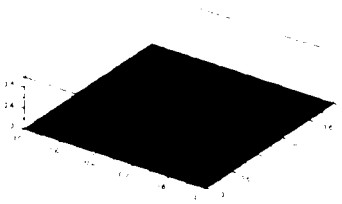


Fig. 103: Filtre liniare - Gauss

Matricial, un exemplu de astfel de filtru are forma:

0	1	1	1	0
1				1
1		9		1
1				1
0	1	1	1	0

Tab. 9: Filtre liniare - Gauss

Punctului central i se atribuie astfel valoarea maximă a ponderii (1.0 în cazul unei reprezentări în numere reale – cu factor de scalare 9), iar punctele învecinate vor contribui la nivelare cu ponderi mai mici.

Exemple de rezultate ale filtrării imaginilor cu filtre de tip 3×3 și 5×5 pentru două valori ale lui σ sunt redate în figurile următoare:

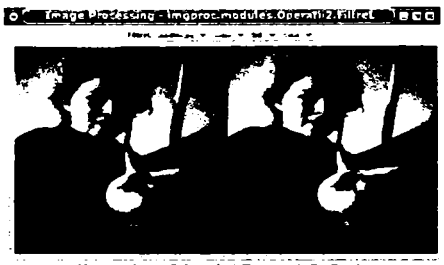


Fig. 106: Filtrul "Gauss" 5x5 (1)

Fig. 107: Filtrul "Gauss" 5x5 (2)

După cum se poate observa, cu creșterea valorii lui σ și sau a dimensiunii matricii filtrului, se obține și o creștere a luminozității imaginii rezultate.

Filtrul diferențial "LoG" (Laplace over Gauss)

Datorită faptului că unele valori ale matricii de filtrare sunt negative, filtrul este denumit și filtru diferențial. Prin aceasta, valorile locale ale imaginii nu vor fi aplatizate, ci din contră diferențele vor fi accentuate.

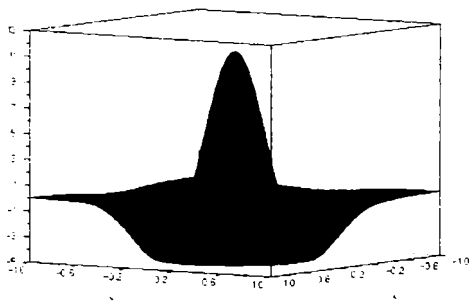


Fig. 108: Filtre liniare - LoG

Reprezentarea matricială este în acest caz:

0	0	-1	0	0
0	-1	-1	-1	0
-1	-1	16	-1	-1
0	-1	-1	-1	0
0	0	-1	0	0

Tab. 10: Filtre liniare – LoG

Figurile următoare prezintă rezultatele filtrării unei imagini cu ajutorul acestui filtru.

Fig. 109: Filtrul "LoG" 3x3

Fig. 110: Filtrul "LoG" 5x5

Proprietăți ale filtrelor liniare

Pentru înțelegerea funcționării filtrelor liniare este însă necesar să facem o incursiune în proprietățile acestora, dat fiind faptul că modul de funcționare al filtrelor nu este o descoperire datorată prelucrării digitale a imaginilor, ci este matematic cunoscut de mult timp.

Convoluția liniară

Convoluția liniară este cunoscută ca fiind operația asupra două funcții discrete I și H sub forma

$$I'(u,v)=\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} I(u-i,v-j)\cdot H(i,j)$$

sau prescurtat

$$I'=I*H$$

Transformând ecuația de mai sus, putem scrie

$$I'(u,v)=\sum_{(i,j)\in R} I(u-i,v-j)\cdot H(i,j)=\sum_{(i,j)\in R} I(u+i,v+j)\cdot H(-i,-j)$$

deci, operația este simetrică atât orizontal cât și vertical.

Operația matematică ce stă la baza tuturor filtrelor este convoluția, notată în continuare cu *.

Liniaritate

Filtrele amintite mai sus sunt denumite liniare tocmai datorită proprietății lor de liniaritate, așadar, prin înmulțirea unei imagini cu o constantă, valoarea convoluției este de asemenea înmulțită cu aceeași constantă. Matematic, acest lucru se poate scrie sub forma:

$$(a\cdot I)*H=I*(a\cdot H)=a\cdot(I*H)$$

De asemenea, convoluția sumei două imagini este echivalentă cu suma convoluțiilor imaginilor inițiale, sau

$$(I_1+I_2)*H=I_1*H+I_2*H$$

pe când rezultatul convoluției sumei cu o constantă a imaginii inițiale nu este egală cu adunarea constantei la rezultatul convoluției

$$(c+I)*H \neq c+I*H$$

Deși liniaritatea filtrelor este un concept matematic foarte important, în realitate, datorită erorilor de calcul și 'clampig'-ului este necesare reducerea liniarității la un minim necesar.

Comutativitate

Convoluția este comutativă, deci

$$I*H=H*I$$

ceea ce ne indică faptul că la urma urmei, orice imagine poate fi utilizată ca filtru și orice filtru ca imagine.

Asociativitate

Prin asociativitate, ordinea de efectuare a operațiilor de convoluție este irelevantă, rezultatul fiind același, lucru foarte important în prelucrarea imaginilor, deoarece, în cazul necesității aplicării mai multor filtre asupra unei imagini inițiale, rezultatul nu depinde de ordinea aplicării filtrelor. Matematic, această proprietate poate fi descrisă îm modul binecunoscut:

$$A*(B*C)=(A*B)*C$$

Separabilitate

Separabilitatea filtrelor este de fapt o proprietate rezultată din cea de asociativitate, dat fiind faptul că oricare dintre imaginile utilizate poate fi considerată ca fiind rezultatul unei convoluții anterioare. Astfel

$$I*H=I*(H)=I*(H_1*H_2*\cdots H_n)=(\cdots(I*H_1)*H_2*\cdots*H_n)$$

Din punct de vedere practic, această proprietate ne permite aplicarea filtrelor în ordinea dorită, astfel încât, în locul unui filtru de dimensiuni mari - care ar necesita o putere de calcul ridicată – vor putea fi utilizate filtre componente de dimensiuni reduse.

Pe de altă parte, este posibilă și separarea filtrelor bidimensionale în două filtre monodimensionale prin descompunerea matricii H în doi vectori H_x și H_y.

Ca urmare, vom obține:

$$I'=(I*H_x)*H_y=I*(H_x*H_y)$$

Așadar, aplicarea celor două filtre monodimensionale este echivalentă cu aplicarea filtrului bidimensional rezultat din convoluția celor două filtre monodimensionale inițiale.

În cazul filtrului Gauss, separarea are loc sub forma:

$$I'=I*G_{\sigma\,x,y}=I*e^{-\frac{x^2+y^2}{2\sigma^2}}=I*(e^{-\frac{x^2}{2\sigma^2}}\cdot e^{-\frac{y^2}{2\sigma^2}})=I*G_{\sigma,x}*G_{\sigma,y}$$

De exemplu, în locul aplicarii a două filtre monodimensionale [0 1 0] și [0 1 2 1 0] se poate utiliza filtrul bidimensional de tip 3x5:

$$[0\,1\,0]*[0\,1\,2\,1\,0]=\begin{bmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 1 & 2 & 1 & 0\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

Filtre neliniare

Deşi foarte importante şi utile, filtrele liniare au însă marele dezavantaj al nivelării punctelor de contur ale imaginii. Astfel, în zonele în care apar puncte izolate, linii sau cante, acestea vor fi de asemenea filtrate şi aplatizate, efect de cele mai multe ori în RP de nedorit. Din acest motiv, vom încerca să eliminăm acest dezavantaj prin utilizarea de filtre neliniare.

Dat fiind faptul că denumirea de 'neliniar' cuprinde tot ceea ce este în afara domeniului 'liniar', numărul şi variaţia filtrelor neliniare sunt nelimitate.

Un alt dezavantaj al acestor filtre este faptul că neavând la bază un model matematic puternic, rezultatele obţinute sunt de cele mai multe ori imprevizibile. De asemenea, ordinea aplicării lor este în general de importanţă crucială şi nu indiferentă, ca în cazul filtrelor liniare. Cu toate acestea, datorită avantajelor deosebite pe care le oferă, aceste filtre sunt indispensabile în vederea prelucrării informaţiilor imaginilor digitale.

Aceste filtre utilizează valori locale din interiorul unei domeniu (regiuni) D ale imaginii iniţiale I.

Filtrul "Miminum"

Notând cu $D_{u,v}$ domeniul din jurul pixelului de coordonate u şi v din imaginea iniţială, filtrul minimum este definit ca

$$I_{min}(D_{u,v})=min\{I'u+i,v+j) \mid i,j\in D\}$$

Prin utilizarea sa, pixelii cu valori ridicate ai regiunii de filtrare vor fi înlocuiţi cu valoarea minimă aflată în regiune. Schematic, principiul de funcţionare al filtrului este prezentat în Fig. 111.

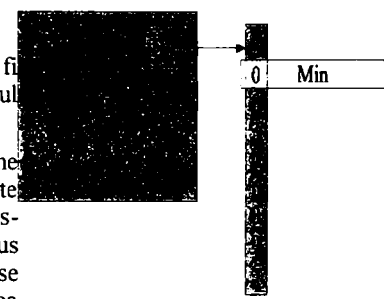


Fig. 111: Filtrul Minimum - principiu

Prin utilizarea sa, valorile mari ale pixelilor din regiunea de imagine analizată vor fi înlocuite cu valorile minime întâlnite. Filtrul este foarte folositor în vederea eliminării 'punctelor albe' din imagini distorsionate. În figurile următoare, în imaginea de referinţă am introdus volutar pixeli şi regiuni de tip 'sare şi piper', deci cu pete luminoase şi în unecate, tocmai pentru a simula distorsiunile amintite şi a putea prezenta mai clar modul de funcţionare a filtrelor.

Utilizând un filtru minim de tip 3×3, se poate cu uşurinţă observa () că punctele albe au fost complet eliminate din imaginea iniţială, în schimb cele negre au fost puternic accentuate.

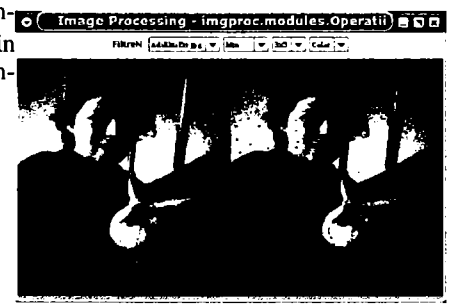


Fig. 112: Filtrul "Minimum" 3x3

Utilizând un filtru de tip 5×5, punctele albe foarte apropiate vor fi și ele eliminate, în schimb accentuarea punctelor întunecate (chiar și a celor dorite) din imagine este deranjantă, atât pentru imagini color

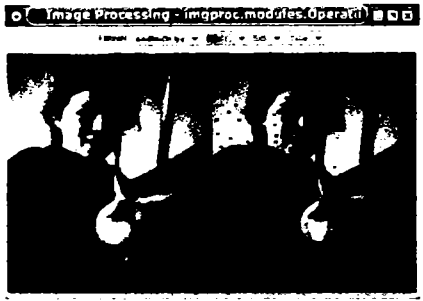


Fig. 113: Filtrul "Minimum" 5x5 color

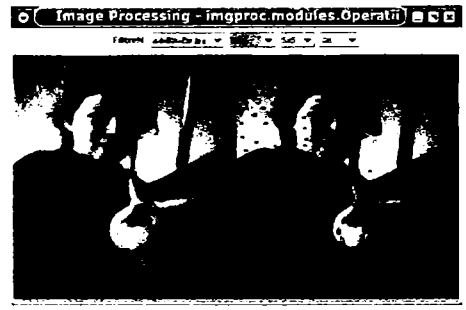


Fig. 114: Filtrul "Minimum" 5x5 gri

cât și în nuanțe de gri.

Filtrul "Maximum"

În principiu este vorba în acest caz de filtrul cu efect invers al minimului, deci

$$I_{max}(D_{u,v}) = max\{ I(u+i, v+j) \mid i, j \in D \}$$

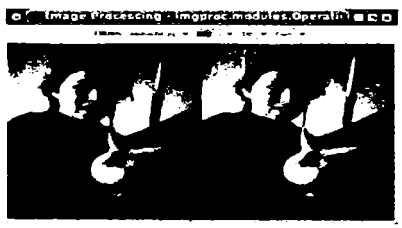


Fig. 116: Filtrul "Maximum" 3x3

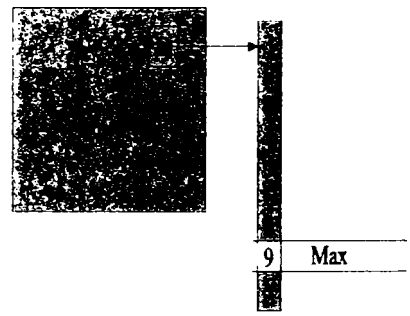


Fig. 115: Filtrul "Maximum" - principiu

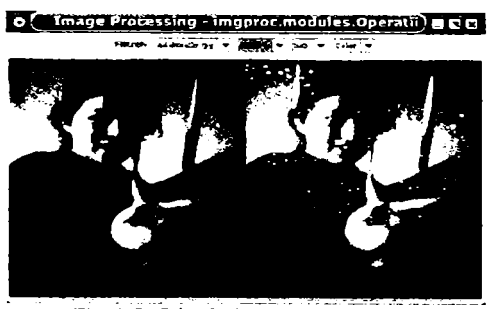


Fig. 117: Filtrul "Maximum" 5x5 color

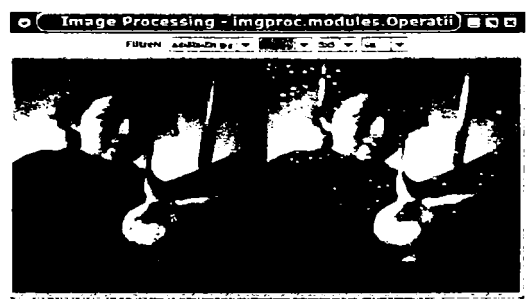


Fig. 118: Filtrul "Maximum" 5x5 gri

Așadar, în acest caz, punctele negre sau întunecate ale imaginii vor fi 'retușate'. Primul exemplu prezintă comportamentul filtrelor de tip 3×3 și 5×5 aplicate asupra aceleiași imagini ca în cazul filtrului anterior.

Filtrul "Median"

Cu toate că este imposibilă formularea unui filtru care să elimine toate defectele și distorsiunile dintr-o imagine digitală și totodată să păstreze informațiile utile, filtrul mediu este cel care se apropie cel mai mult de acest deziderat, utilizând valoarea medie pixelilor din interiorul regiunii analizate.

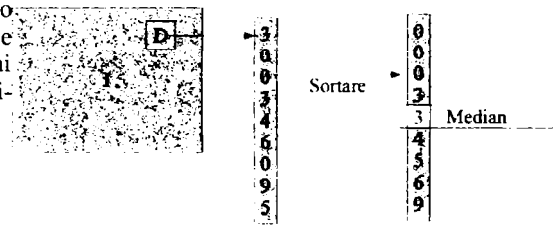


Fig. 119: Filtrul median - principiu

$$I_{med}(D_{u,v}) = median\{ I(u+i, v+j) \mid i, j \in D \}$$

Cu $median$ am notat valoarea aflată la mijlocul filtrului, după algoritmul prezentat schematic în figura alăturată:

După cum se poate observa din figurile următoare, prin aplicarea acestui tip de filtre, calitatea imaginii rezultate este satisfăcătoare, chiar și în cazul filtrelor de tip 5×5. Pentru cazul filtrului de tip 3×3 distorsiunile au fost complet eliminate, iar accentuarea nivelelor de culoare intunecate sau luminoase nu are loc.

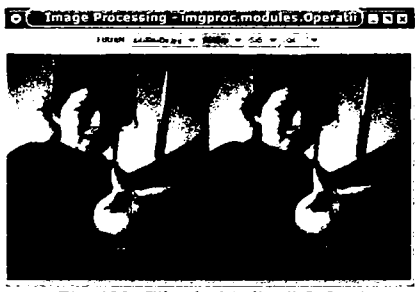


Fig. 120: Filtrul "Median" 5x5 gri

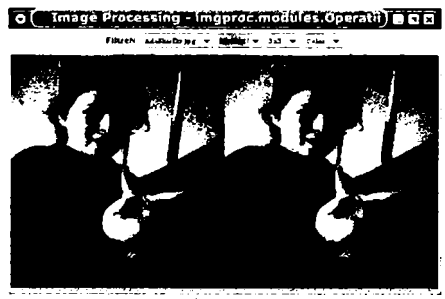


Fig. 121: Filtrul "Median" 3x3

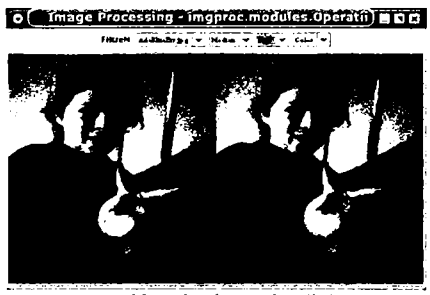


Fig. 122: Filtrul "Median" 5x5

5.2.2.5 Tehnici de analiză spectrală

O metodă foarte răspândită de analiză a problemelor din cele mai diverse domenii ale activității umane este cea a analizei domeniilor de frecvențe. Metodele de analiză prezentate în continuare și-au câștigat un loc principal în cele mai diverse dintre aceste domenii putând fi întâlnite de exemplu începând cu domeniul sistemelor liniare, în optică, antene și sisteme stocastice, până la domenii ale fizicii quantice.

Utilizarea transformărilor Fourier de exemplu, este nu numai o teorie interesantă, ci extinde într-un mod foarte elegant domeniile prezentate anterior referitoare la filtre liniare.

Transformarea Fourier

Idea transformării informațiilor prin separarea lor în vibrații sinusoidale pure, provenită inițial în urma analizării sistemelor acustice (ton, sunet, muzică), a fost în ultimul timp tot mai mult extinsă asupra altor domenii, dat fiind faptul că în realitate, vibrațiile sinusoidale apar în toate tipurile de oscilații naturale existente.

Analiza și sinteza

După cum se poate observa din figura următoare, semnalul de culoare roșie, este suma celorlalte trei semnale.

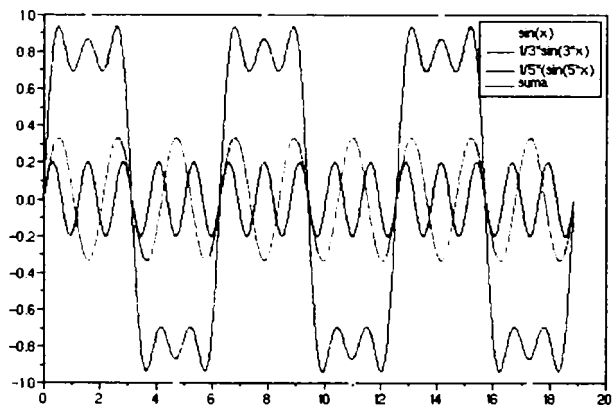


Fig. 123 Semnal periodic rezultat din suma a trei semnale oscilante

Așadar, semnalul rezultat este dat de formula:

$$s(x)=s_1(x)+s_2(x)+s_3(x)=\sum_{n=1}^{3}\frac{1}{2n-1}\sin((2n-1)x)$$

sau ținând cont de faptul că funcția $\sin(x)$ este periodică, pentru o frecvență de bază f_0 vom avea:

$$s(x)=s_1(x)+s_2(x)+s_3(x)=\sum_{n=1}^{3}\frac{1}{2n-1}\sin(2\ \pi(2n-1)f_0\,x)$$

Reprezentarea în domeniul de valori

În domeniul de valori, reprezentarea funcţiei este cea însumată, deci utilizând exemplul de mai sus, reprezentarea va fi ca în figura următoare:

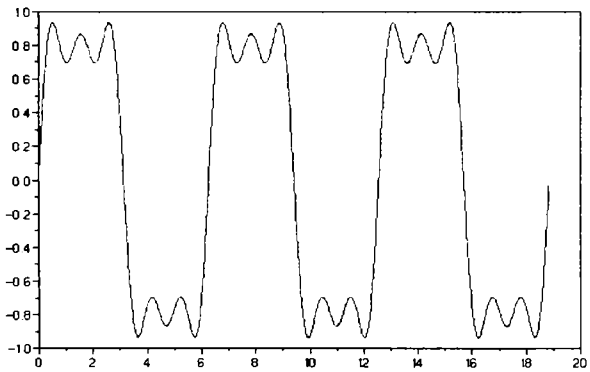


Fig. 124: Fourier - Reprezentarea în domeniul de valori

Reprezentarea în domeniul de frecvenţă

Corespunzător frecvenţelor componente, în acest caz vor fi reprezentate frecvenţele şi amplitudinile semnalelor componente, ca în firuga ce urmează:

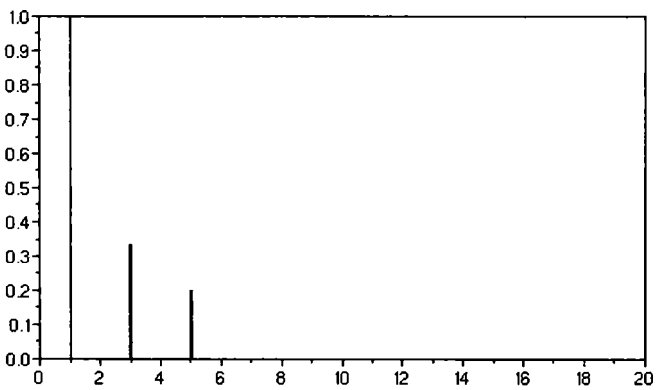


Fig. 125: Fourier - Reprezentarea în domeniul de frecvenţă

Serii Fourier

Notând cu P_0 perioada de bază a semnalului $(P_0 = \frac{1}{f_0})$ şi considerând la limită că $n \to \infty$ vom obţine forma generală a unei serii Fourier:

$$s(x)=\sum_{n=1}^{x}\frac{1}{2n-1}\sin\frac{2\pi(2n-1)x}{X_0}$$

Ţinând cont de faptul că funcţia este periodică, este suficientă analiza comportamentului acesteia în interiorul unei perioade X_0, deoarece

$$s(x)=s(x+kX_0)$$

Aşadar, va fi necesară şi suficientă analiza unui şir de elemente de forma $\sin(\frac{2\pi k}{X_0}x)$ şi

$\cos(\frac{2\pi k}{X_0}x)$ în care k este un număr întreg opţional. Procesul este numit 'Analiză armonică', iar şirul rezultat este o 'serie Fourier'.

Serii Fourier în notaţia lui Euler

O altă notaţie importantă, este cea dată de Euler, în care analiza armonică are loc în spaţiul complex. Pornind de la idea că un număr complex poate fi scris sub forma

$$z=e^{i\Theta}=\cos(\Theta)+i\cdot\sin(\Theta)$$

cercul trigonometric va fi dat de

$$z=e^{i\omega_0 x}$$

şi deci vom putea nota

$$e^{i\omega_0 x}=\cos(\omega_0 x)+i\cdot\sin(\omega_0 x)$$

unde ω_0 este frecvenţa oscilaţiei.

În cazurile de mai sus, fără a îngrădi generalitatea, am considerat amplitudinile funcţiilor componente ca fiind egale iar fazele lor iniţiale ca fiind egale cu 0. În mod normal, ecuaţia unei funcţii periodice fiind dată de [22]

$$s(x)=A\sin(\omega x+\varphi)$$

În exemplul următor este demonstrată o apropiere de curba 'dinte de fierăstrău' prin utilizarea suprapunerii funţiilor periodice [23]

$$y=A_0+A_1\sin(\omega t)+A_2\sin(2\omega t)+\ldots$$

pentru cazul special

$$y=-\sin(\omega t)-\frac{1}{2}\sin(2\omega t)-\frac{1}{3}\sin(3\omega t)-\frac{1}{4}\sin(4\omega t)-\frac{1}{5}\sin(5\omega t)$$

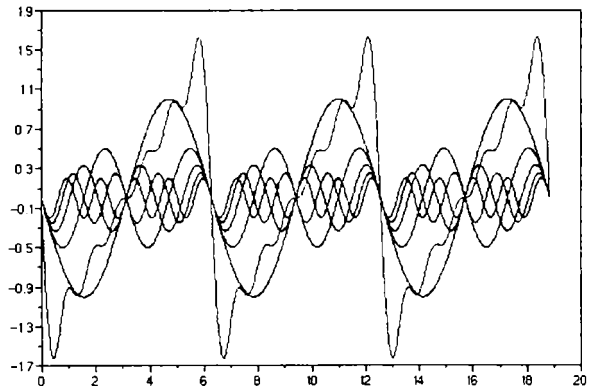


Fig. 126: Fourier - Aproximarea funcției 'dinte de fierăstrău'

Ținând cont de amplitudini, frecvențe și faze, forma generală a seriilor Fourier poate fi astfel scrisă sub forma:

$$g(x)=\sum_{k=0}^{\infty}\left[A_k\cos(k\omega_0 x)+B_k\sin(k\omega_0 x)\right]$$

în care A_k și B_k poartă denumirea de coeficienți Fourier

Integrala și spectrul Fourier

Fourier însă a fost de părere că acest concept este aplicabil și pentru funcții neperiodice, postulând faptul că și acestea pot fi determinate prin sume ale funcțiilor în sinus și cosinus. Astfel, s-a ajuns la conceptul de integrală Fourier, dată de formula:

$$g(x)=\int_0^{\infty}(A_\omega\cos(\omega x)+B_\omega\sin(\omega x))\,d\omega$$

Această integrală este elementul de bază atât al spectrului cât și a transformărilor Fourier. Fiecare dintre coeficienții A_ω și B_ω specifică valoarea amplitudinii cu care funcțiile sinus și cosinus corespondente participă la formarea semnalului rezultat $g(x)$. Întrebarea esențială care rezultă este însă "este posibilă determinarea unică acestor coeficienți ?". Răspunsul este evident "Da", iar demonstrația este relativ simplă.

Notând cu

$$A_\omega=A(\omega)=\frac{1}{\pi}\int_{-\infty}^{\infty}g(x)\cdot\cos(\omega x)\,dx$$

și

$$B_\omega=B(\omega)=\frac{1}{\pi}\int_{-\infty}^{\infty}g(x)\cdot\sin(\omega x)\,dx$$

ponderile amintite, dat fiind faptul că ω aparţine unui domeniu de frecvenţe continue, este evident faptul că atât A_ω cât şi B_ω vor fi de asemenea continui. Ele vor conţine un spectru de componente de frecvenţă ale semnalului original $g(x)$. Integrala Fourier descria astfel funcţia iniţială ca fiind suma unui număr infinit de funcţii sinus şi cosinus ale căror coeficienţi sunt daţi de A_ω şi B_ω. Ultimele două relaţii prezintă astfel modul de obţinere al spectrului corespunzător, iar relaţia anterioară ne arată cum putem obţine funcţia dorită utilizând spectrul determinat.

Transformarea Fourier

În urma determinării coeficienţilor A_ω şi B_ω suntem la un pas minimal de aşa numita transformare Fourier. În cazul transformării Fourier, atât integrala cât şi spectrul sunt considerate ca fiind funcţii complexe. Pornind de la cele două reprezentări ale coeficienţilor, vom defini spectrul Fourier $G(\omega)$ al unei funcţii $g(x)$ ca fiind

$$\begin{aligned} G(\omega) &= \sqrt{\left(\frac{\pi}{2}\right)}\left[A(\omega) - i\cdot B(\omega)\right] \\ &= \sqrt{\left(\frac{\pi}{2}\right)}\left[\frac{1}{\pi}\int_{-\infty}^{\infty} g(x)\cdot\cos(\omega x)\,dx - i\cdot\frac{1}{\pi}\int_{-\infty}^{\infty} g(x)\cdot\sin(\omega x)\,dx\right] \\ &= \frac{1}{\sqrt{(2\pi)}}\int_{-\infty}^{\infty} g(x)\cdot\left[\cos(\omega x) - i\cdot\sin(\omega x)\right]dx \\ &= \frac{1}{\sqrt{(2\pi)}}\int_{-\infty}^{\infty} g(x)\cdot e^{-i\omega x}\,dx \end{aligned}$$

Trecerea de la funcţia $g(x)$ la spectrul ei $G(\omega)$ poartă denumirea de "Transformare Fourier" iar transformarea prin care, de la spectrul $G(\omega)$ se trece la obţinerea funcţiei $g(x)$, poartă numele de "Transformarea Fourier inversă", dată de relaţia

$$g(x) = \frac{1}{\sqrt{(2\pi)}}\int_{-\infty}^{\infty} G(\omega)\cdot e^{i\omega x}\,d\omega$$

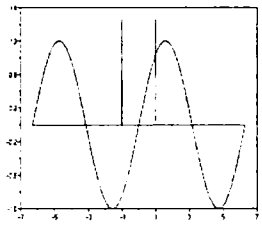
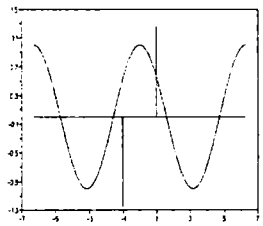
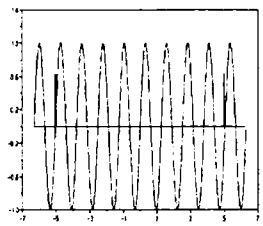
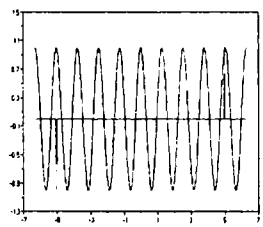
După cum se observă, transformarea directă şi cea inversă sunt simetrice.

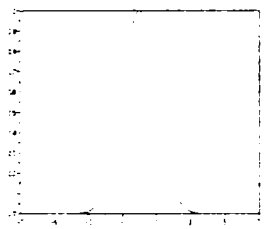
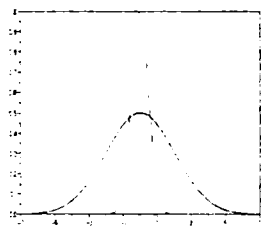
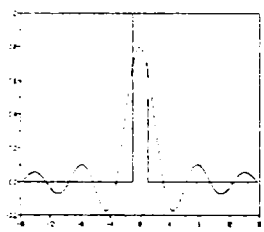
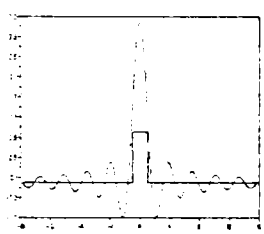
Perechi de transformări Fourier

Date fiind faptul că între funcţia $g(x)$ şi spectrul ei $G(\omega)$ există o relaţie de determinare reciprocă unică în ambele sensuri, ambele funcţii pot fi grupate ca pereche sub notaţia

$$g(x) \multimap G(\omega)$$

În tabelul de mai jos sunt prezentate unele perechi de transformări Fourier ale unor funcţii analitice reprezentante

$g(x)$	***Grafic*** $g(x)$ + $G(\omega)$	$G(\omega)$
$g(x)=\sin(x)$		$G(\omega)=\sqrt{\frac{\pi}{2}}\cdot(\delta(\omega-1)+\delta(\omega+1))$
$g(x)=\cos(x)$		$G(\omega)=i\sqrt{\frac{\pi}{2}}\cdot(\delta(\omega-1)-\delta(\omega+1))$
$g(x)=\sin(5x)$		$G(\omega)=\sqrt{\frac{\pi}{2}}\cdot(\delta(\omega-5)+\delta(\omega+5))$
$g(x)=\cos(5x)$		$G(\omega)=i\sqrt{\frac{\pi}{2}}\cdot(\delta(\omega-5)-\delta(\omega+5))$

$g(x)$	Grafic $g(x)$ + $G(\omega)$	$G(\omega)$
$g(x)=e^{-\frac{x^2}{2}}$		$G(\omega)=e^{-\frac{\omega^2}{2}}$
$g(x)=\frac{1}{2}e^{-\frac{x^2}{2\cdot 2^2}}$		$G(\omega)=e^{-\frac{2^2\omega^2}{2}}$
$g(x)=\Pi_1(x)$		$G(\omega)=\frac{2\sin(\omega)}{\sqrt{2\pi}\omega}$
$g(x)=\Pi_2(x)$		$G(\omega)=\frac{2\cdot 2\sin(2\omega)}{\sqrt{2\pi}\omega}$

Din tabelul anterior, putem observa în primul rând că perechea unei funcţii Gauss este tot o funcţie Gauss, în cazul în care $\sigma=1$ cele două funcţii sunt chiar identice. În cazul unei funcţii dilatate Gauss, perechea ei este o funcţie comprimată, şi invers. În cazul funcţiilor sinus şi cosinus, perechea transformării este obţinută prin intermediul unei funcţii Dirac (impus), iar perechea unei funcţii de tip impuls dreptunghiular

este o funcție de tip $\sin\frac{(x)}{x}$.

Proprietăți ale transformărilor Fourier

În continuare vom aminti cele mai importante proprietăți ale transformărilor Fourier.

Liniaritatea

Transformarea Fourier este o operație liniară, așadar

$$a \cdot g(x) \multimap a \cdot G(\omega), \forall a \in \mathbb{C}$$

și

$$g_1(x) + g_2(x) \multimap G_1(\omega) + G_2(\omega)$$

În cazul semnalelor bidimensionale vom avea

$$g_1(x,y) + g_2(x,y) \multimap G_1(\omega_x, \omega_y) + G_2(\omega_x, \omega_y)$$

Ca exemplu, în Fig. 127 este reprezentată o imagine simplă care în Fig. 128 a fost micșorată. După cum se observă din Fig. 129 și 130 - spectrele corespunzătoare acestor imagini, pentru imaginea micșorată s-a obținut un spectru mărit.

Fig. 127: FFT Liniaritate 1

Fig. 128: FFT Liniaritate 2

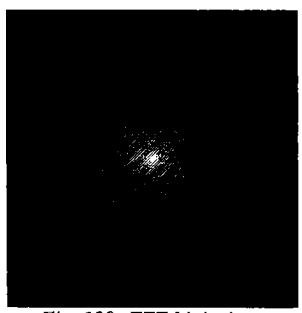


Fig. 129: FFT Liniaritate spectru 1

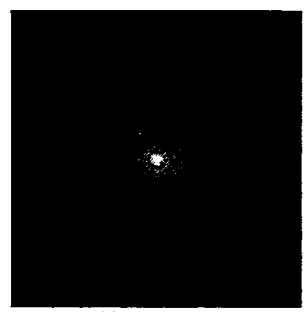


Fig. 130: FFT Liniaritate spectru 2

Cazul superpoziţiei, deci a sumei a două semnale, este prezentat în imaginile următoare.

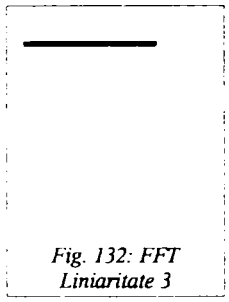


Fig. 132: FFT Liniaritate 3

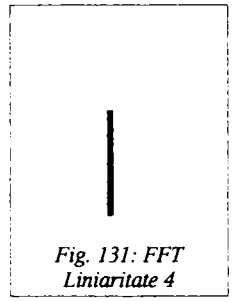


Fig. 131: FFT Liniaritate 4

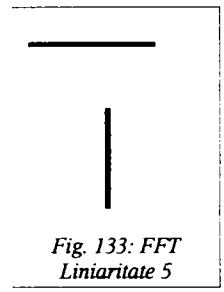


Fig. 133: FFT Liniaritate 5

Fig. 134: FFT Liniaritate spectru 3

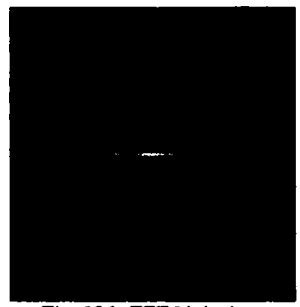


Fig. 136: FFT Liniaritate spectru 4

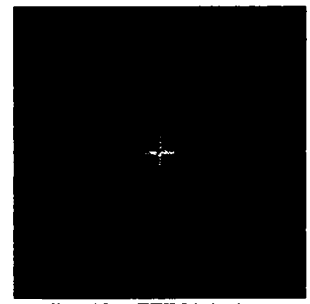


Fig. 135: FFT Liniaritate spectru 5

Deplasabilitate

Prin deplasarea funcţiei cu o distanţă d de-a lungul axei de coordonate va atrage cu sine multiplicarea spectrului Fourier corespunzător cu un factor complex al lui ω de valoare $e^{-i\omega d}$

$$g(x-d) \multimap e^{-i\omega d} \cdot G(\omega)$$

Asemănător, în cazul bidimensional, vom avea

$$g(x-d_x, y-d_y) \multimap G(\omega_x, \omega_y) \cdot e^{-i2\pi(d_x\omega_x + d_y\omega_y)}$$

şi

$$g(x,y)e^{i2\pi(x\omega_x + y\omega_y)} \multimap G(\omega_x - D_x, \omega_y - D_y)$$

Această proprietate are repercursiuni de importanţă majoră pentru analiza imaginilor, dat fiind faptul că senzorii externi (indiferent dacă este vorba de o cameră digitală sau ochiul uman) sesizează doar intensităţi, neputând înregistra stimulaţii complexe.

După cum se observă din relaţiile de mai sus, în spectrul de frecvenţe nu apare nici o transformare, amplitudinile imaginilor fiind identice şi centrate în jurul originii şi deci în momentul trecerii de la spectrul de valori la spectrul de frecvenţe, poziţia (deci deplasarea) este ignorată.

Imaginile următoare exemplifică această proprietate.

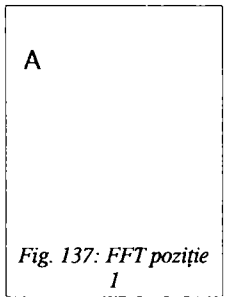


Fig. 137: FFT poziţie 1

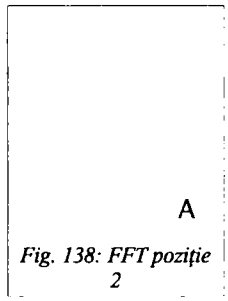


Fig. 138: FFT poziţie 2

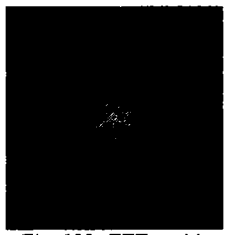


Fig. 139: FFT poziţie spectru 1

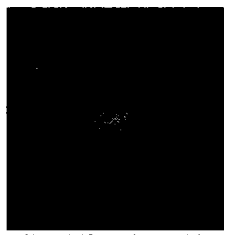


Fig. 140: FFT poziţie spectru 2

Simetria (Rotativitatea)

Spectrul Fourier se întinde peste frecvenţele pozitive şi negative putând avea şi valori complexe. Însă ele sunt simetrice faţă de origine

$$G(\omega)=G^*(-\omega)$$

unde cu G^* am notat conjugata complexă a lui G.

În cazul efectuării unei rotaţii în spectrul valorilor cu un unghi α se va obţine în spectrul frecvenţelor o rotaţie cu acelaşi unghi

$$g(x\cos\alpha+y\sin\alpha,-x\sin\alpha+y\cos\alpha)\multimap G(\omega_x\cos\alpha+\omega_y\sin\alpha,-\omega_x\sin\alpha+\omega_y\cos\alpha)$$

Prin rotirea cu 90° a imaginii din Fig. 137 vom obţine, spre exemplificare, imaginea şi spectrul ei corespunzător conform figurilor următoare.

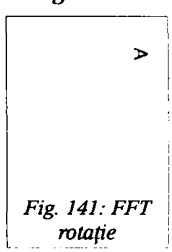


Fig. 141: FFT rotaţie

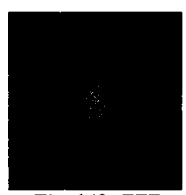


Fig. 142: FFT rotaţie spectru

Asemănarea

Scalarea în timp sau spaţiu a funcţiei $g(x)$ atrage cu sine scalarea spectrului

$$g(sx)\multimap\frac{1}{|s|}\cdot G(\frac{\omega}{s})$$

cazul bidimensional fiind dat de

$$g(s_x x, s_y y) \multimap \frac{1}{|s_x s_y|} G\left(\frac{\omega_x}{s_x}, \frac{\omega_y}{s_y}\right)$$

Din relaţia de mai sus, reiese faptul că o expandare a domeniului valorilor după o axă atrage cu sine o aplatizare a funcţiilor de frecvenţă corespunzătoare.

Fin figurile următoare reiese cu uşurinţă faptul că, în momentul scalării imaginii are loc şi scalarea spectrului prin aplatizarea funcţiilor de frecvenţă.

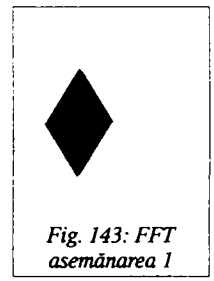


Fig. 143: FFT asemănarea 1

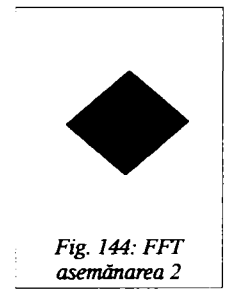


Fig. 144: FFT asemănarea 2

Fig. 145: FFT asemănarea spectru 1

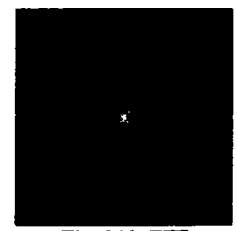


Fig. 146: FFT asemănarea spectru 2

Suprapunerea

Cel mai important aspect al transformărilor Fourier este însă pentru cazul nostru raportul în care acestea stau cu suprapunerea liniară. Luând în considerare două funcţii $g(x)$ şi $h(x)$ împreună cu spectrele lor $G(\omega)$ şi $H(\omega)$ şi supunând cele două funcţii unei suprapuneri liniare $g(x) * h(x)$, vom obţine următoarea relaţie:

$$g(x) * h(x) \multimap G(\omega) \cdot H(\omega)$$

deci. prin suprapunerea valorilor celor două funcţii obţinem produsul spectrelor (punct cu punct).

Datorită dualităţii dintre spaţiul local la funcţiilor şi domeniul lor spectral, relaţia este valabilă şi în sens invers, deci:

$$g(x) \cdot h(x) \multimap G(\omega) * H(\omega)$$

aşadar, multiplicarea semnalelor atrage cu sine suprapunerea spectrelor.

Notând cu $\ddagger$ suprapunerea bidimensională, vom avea în acest caz:

$$g_1(x, y) \ddagger g_2(x, y) \multimap G_1(\omega_x, \omega_y) G_2(\omega_x, \omega_y)$$

şi

$$g_1(x, y)\, g_2(x, y) \multimap G_1(\omega_x, \omega_y) \ddagger G_2(\omega_x, \omega_y)$$

În imaginile următoare vom exemplifica suprapunerea spectrelor.

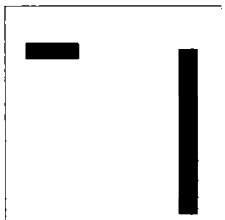


Fig. 149: FFT suprapunerea 1

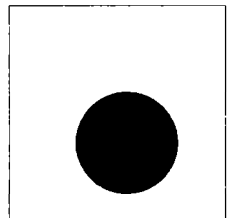


Fig. 148: FFT suprapunerea 2

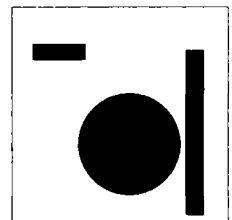


Fig. 147: FFT suprapunerea 3

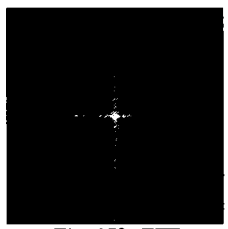


Fig. 152: FFT suprapunerea spectru 1

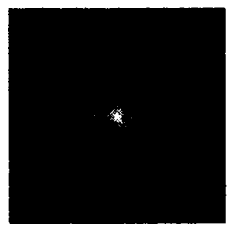


Fig. 151: FFT suprapunerea spectru 2

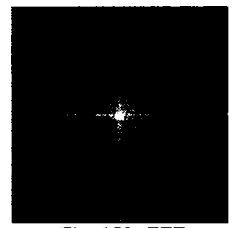


Fig. 150: FFT suprapunerea spectru 3

Trecerea la valori discrete

Din nefericire, utilizarea domeniilor continue pe calculatoare numerice nu este posibilă, fapt pentru care suntem nevoiţi să reducem acest domeniu la valori discrete. Metoda principală utilizată este cea de alegere a valorilor necesare în formă vectorială, atât a valorilor cât şi a spectrului transformării. Procedura este numită 'sampling' şi o vom prezenta în continuare.

Sampling

Prin sampling se înţelege extragerea unor valori discrete dintr-o funcţie continuă în anumite puncte din spaţiu şi/sau timp, în general aflate la intervale regulate.

Pentru exprimarea matematică formală avem nevoie de noţiunea de funcţie impuls (cunoscută şi sub numele de funţie Delta sau Dirac). Această funcţie este deosebită, prin faptul că toate valorile ei sunt nule, cu excepţia originii unde valoarea sa este nenulă însă nedefinită iar integrala sa nedefinită are valoarea 1.

Această funcţie, poate fi matematic notată prin expresia

$$\delta(x)=\begin{cases} 0 \quad pentru\, x\neq 0 \\ \int_{-\infty}^{\infty} \delta(x)dx=1 \; pentru\, x=0 \end{cases}$$

Această funcţie poate fi imaginată ca fiind reprezentată de un singur puls de lăţime infinit mică dar a cărui energie este finită şi egală cu 1. Datorită acestui fapt, această funcţie nu poate fi reprezentată (în imaginile pe care le utilizăm este vorba de o aproximare a acestei funcţii în vederea obţinerii unei claritâţi mai ridicate a explicaţiilor).

O altă caracteristică esenţială a aceste funcţii este legată de scalarea acestei funcţii, dat fiind faptul că inte-

grala sa nedefinită trebuie să rămână de valoare 1, şi deci

$$\delta(sx)=\frac{1}{|s|}\delta(x),\forall s\neq 0$$

Sampling (tastare) cu funcţii impuls

În cazul în care o funcţie continuă $g(x)$ este supusă unui sampling prin intermediul unei funcţii impuls $\delta(x)$, vom obţine o nouă funcţie de forma

$$\overline{g}(x)=g(x)\cdot\delta(x)=\begin{cases} g(0)\ pentru\ x=0 \\ 0 \quad altfel \end{cases}$$

Prin deplasare originii funcţiei impuls, vom putea astfel tasta orice punct din domeniul de definiţie a funcţiei de analizat.

$$\overline{g}(x)=g(x)\cdot\delta(x-x_0)=\begin{cases} g(x_0)\ pentru\ x=x_0 \\ 0 \quad altfel \end{cases}$$

sau, dat fiind faptul că în cele mai multe cazuri avem nevoie de o secvenţă de puncte de tastat, putem defini funcţia necesară ca fiind suma funcţiilor impuls componente

$$\overline{g}(x)=g(x)\cdot\delta(x-x_0)+g(x)\cdot\delta(x-x_1)+\ldots=g(x)[\delta(x-x_0)+\delta(x-x_1)+\ldots]$$

În cazul unei tastări a N puncte, formula necesară poate fi scrisă simplificat sub forma

$$\overline{g}(x)=g(x)\cdot\sum_{i=1}^{N}\delta(x-x_i)$$

Funcţia piaptăn

Notând suma impulsurilor deplasate ale formulei anterioare în modul următor

$$III(x)=\sum_{i=1}^{N}\delta(x-x_i)$$

vom putea rescrie ecuaţia anterioară sub forma

$$\overline{g}(x)=g(x)\cdot III(x)$$

În relaţia anterioară am considerat intervalul de tastare ca fiind egal cu 1, ceea ce nu este totdeauna satisfăcător. Pentru valori ale intervalului diferite (notate de exemplu cu τ), relaţia de mai sus devine

$$\overline{g}(x)=g(x)\cdot III(\frac{x}{\tau})$$

Repercursiunile tastării asupra spectrului Fourier

Asemănător funcției Gauss, funcția piaptăn are proprietatea că transformata sa Fourier este tot o funcție de tip piaptăn, deci

$$III(x)\multimap III(\frac{1}{2\pi}\omega)$$

sau, în cazul unei sclări a domeniului de tastare

$$III(\frac{x}{\tau})\multimap \tau\, III(\frac{\tau}{2\pi}\omega)$$

Folosindu-ne de proprietatea de suprapunere a transformărilor Fourier, putem de asemenea determina rezultatul discretizării unui semnal $g(x)$ prin aplicarea funcției piaptăn, deoarece

$$g(x)\cdot III(\frac{x}{\tau})\multimap G(\omega)*\tau\, III(\frac{\tau}{2\pi}\omega)$$

Astfel, prin suprapunerea unei funcții arbitrare cu un impuls $\delta(x)$ vom obține funcția inițială, sau matematic notat

$$f(x)*\delta(x)=f(x)$$

iar suprapunerea unui impuls decalat $\delta(x-d)$ va reproduce funcția inițială deplasată cu aceeași distanță

$$f(x)*\delta(x-d)=f(x-d)$$

Acest lucru conduce la concluzia că spectrul Fourier al semnalului tastat $(\overline{G}(\omega))$ va fi regăsit spectrul semnalului continuu original $(G(\omega))$ replicat la infinit, motiv pentru care spectrul Fourier rezultat este periodic, cu perioada $\frac{2\pi}{\tau}$.

Teoria tastării (sampling theory) și noțiunea de alias

Dacă valorile spectrului obținut $(\overline{G}(\omega))$ nu se interpătrund, spectrul original $(G(\omega))$ poate fi refăcut din orice replicat (și odată cu el deci și funcția $g(x)$), operație care are loc fără pierdere. Din acest motiv, este evident că frecvențele semnalului original $g(x)$ sunt mărginite superior, semnalul trebuind să conțină frecvențe mai mici decât o anumită frecvență ω_{max}.

Dependența dintre frecvența de sampling și cea maximă admisă este astfel dată de

$$\omega_{max}\leq\frac{1}{2}\omega_s$$

sau

$$\omega_s \geq 2\ \omega_{max}$$

deci, pentru reuşita garantată a unui sampling este necesar ca frecvenţa acestuia să fie cel puţin dublă faţă de frecvenţa maximă a semnalului $g(x)$.

În caz contrar, după cum se poate observa din figura următoare, semnalul nu mai poate fi reconstruit fără erori. Acest fenomen poartă denumirea de 'aliasing'.

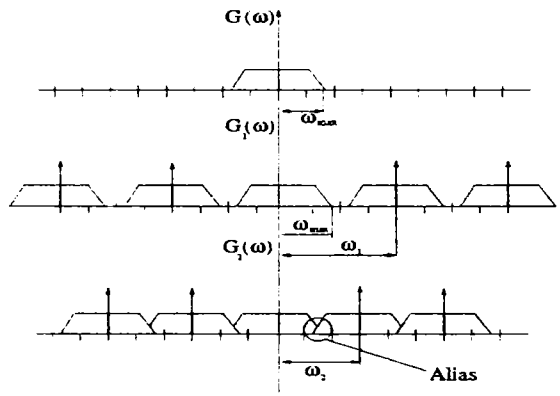


Fig. 153: Fourier aliasing

Transformarea discretă Fourier

Utilizând informaţiile prezentate anterior în cadrul tastării (sampling), putem trece de la transformările Fourier de tip continuu prezentate, la transformări discrete, care poartă denumirea de DFT (Discrete Fourier Transformation), această metodă fiind adecvată implementării pe calculatoare numerice.

Monodimensională

Deşi transformarea discretă Fourier monodimensională pentru prelucrarea digitală a imaginilor nu prezintă o importanţă deosebită, ea stând la baza transformării bidimensionale, o vom descrie pe scurt.

Definiţie

Transformarea discretă Fourier este, la fel ca şi cea continuă, identică în ambele sensuri. Astfel, transformarea directă a unui semnal $g(x)$ de lungime $M, (x=0 \ldots M-1)$ are forma

$$G(m)=\frac{1}{\sqrt{(M)}}\sum_{x=0}^{M-1} g(x)\cdot e^{-i2\pi\frac{mx}{M}}, \text{pentru } 0 \leq m \leq M$$

iar transformarea inversă (DFT^{-1}) este dată de

$$g(x)=\frac{1}{\sqrt{(M)}}\sum_{m=0}^{M-1} G(m)\cdot e^{i2\pi\frac{mx}{M}}, \text{pentru } 0 \leq x \leq M$$

unde atât semnalul g(x) cât şi spectrul G(m) sunt vectori de lungime M, având valori complexe, aşadar

$$G(m)=G_{\Re}(m)+iG_{\Im}(m)$$
$$g(x)=g_{\Re}(x)+ig_{\Im}(x) \qquad \forall x,m=0\cdots M-1$$

Rescriind aceste ecuaţii în sinus şi cosinus, vom obţine

$$G(m)=\frac{1}{\sqrt{(M)}}\sum_{x=0}^{M-1}\underbrace{[g_{\Re}(x)+ig_{\Im}(x)]}_{g(x)}\cdot[\underbrace{\cos(2\pi\frac{mx}{M})}_{C_m^M(u)}-\underbrace{i\cdot\sin(2\pi\frac{mx}{M})}_{S_m^M(u)}]$$

în care C_m^M şi S_m^M reprezintă funcţiile discrete de bază sinus şi cosinus prezentate în continuare.

Astfel, părţile reale şi imaginare ale spectrului Fourier vor putea fi scrise sub forma:

$$G_{\Re}(m)=\frac{1}{\sqrt{(M)}}\sum_{x=0}^{M-1}(g_{\Re}(x)\cdot C_m^M(x)+g_{\Im}(x)\cdot S_m^M(x))$$
$$G_{\Im}(m)=\frac{1}{\sqrt{(M)}}\sum_{x=0}^{M-1}(g_{\Im}(x)\cdot C_m^M(x)-g_{\Re}(x)\cdot S_m^M(x))$$

iar ale transformatei inverse

$$g_{\Re}(x)=\frac{1}{\sqrt{(M)}}\sum_{m=0}^{M-1}(G_{\Re}(x)\cdot C_m^M(x)+G_{\Im}(x)\cdot S_m^M(x))$$
$$g_{\Im}(x)=\frac{1}{\sqrt{(M)}}\sum_{m=0}^{M-1}(G_{\Im}(x)\cdot C_m^M(x)-G_{\Re}(x)\cdot S_m^M(x))$$

Funcţiile discrete de bază

După cum am văzut anterior, transformarea discretă Fourier descrie descompunerea unei funcţii discrete într-o sumă finită de funcţii sinus şi cosinus $(C_m^M \text{ şi } S_m^M)$ ale căror ponderi (amplitudini) sunt determinate de coeficienţii $G(m)$. Fiecare dintre aceste funcţii reprezintă o funcţie sinus, respectiv cosinus, de frecvenţă m pe o perioadă M evaluate într-un punct opţional x.

Aceste funcţii sunt reprezentate după cum urmează:

$$C_m^M(x)=C_x^M=\cos(2\pi\frac{mx}{M})$$
$$S_m^M(x)=S_x^M=\sin(2\pi\frac{mx}{M})$$

Unităţi ale domeniului vabrilor şi a celui spectral

Raportul dintre unităţile celor două domenii mai sus amintite precum şi modul de interpretare al numărului de unde m, duce de foarte multe ori la neînţelegeri.

Orice coeficient spectral $G(m)$ corespunde unei perechi de funcţii sinus şi cosinus având o anumită frecvenţă în domeniul valorilor. Cu toate că atât funcţia cât şi transformata sunt reprezentate de date vecto-

riale fără unităţi de măsură, este necesar să înţelegem în ce relaţie se găsesc axele de coordonate ale spectrului în raport cu cele ale sistemului real.

Considerând un semnal continuu care va fi tastat în M poziţii consecutive la un interval τ (unitate de măsură sau timp), putem face următoarele constatări.

Numărul de undă $m=1$ reprezintă unitatea de bază a semnalului periodic de perioadă $M\tau$ şi având deci frecvenţa

$$f_1=\frac{1}{M\tau}$$

Astfel, frecvenţa unui spectru discret cu numărul de unde m este dată de

$$f_m=m\frac{1}{M\tau}=m\cdot f_1, \quad pentru\ 0\le m<M$$

sau pentru frecvenţe ciclice

$$\omega_m=2\pi f_m=m\frac{2\pi}{M\tau}=m\cdot\omega_1$$

Frecvenţa de sampling este bineânţeles dată de formula

$$f_s=\frac{1}{\tau}=M f_1$$

şi este evident obţinută pentru numărul de undă $m_s=M.$

Astfel, putem determina numărul maxim de unde care pot apărea în cadrul spectrului discretizat fără apariţia fenomenului de alias

$$m_{max}=\frac{M}{2}=\frac{m_s}{2}$$

deci, după cum era de aşteptat, limita este dată de jumătatea numărului de unde ale semnalului de sampling.

Bidimensională (DFT 2D)

Dat fiind faptul că transformarea discretă Fourier nu este legată de dimensiunea domeniului de definiţie al semnalului, transformări de acest tip există teoretic pentru toate dimensiunile şi deci, cu atât mai mult pentru cazul care ne interesează, 2D în care se află imaginile digitale de analizat.

Definiţie

Pentru o funcţie periodică bidimensională $g(u,v)$ de coordonate $u=0\ldots M-1$ şi $v=0\ldots N-1,$ transformarea discretă Fourier poate fi scrisă sub forma

$$G(m,n)=\frac{1}{\sqrt{MN}}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} g(u,v)\cdot e^{-i2\pi\frac{mu}{M}}\cdot e^{-i2\pi\frac{nv}{N}}$$
$$=\frac{1}{\sqrt{MN}}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} g(u,v)\cdot e^{-i2\pi(\frac{mu}{M}+\frac{nv}{N})}$$

şi este aşadar tot o funcţie bidimensională de dimensiune $M\times N$, de coordonate spectrale $m=0\ldots M-1,\ n=0\ldots N-1,$ al cărei inverse este

$$g(u,v)=\frac{1}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} G(m,n)\cdot e^{i2\pi\frac{mu}{M}}\cdot e^{i2\pi\frac{nv}{N}}$$
$$=\frac{1}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} G(m,n)\cdot e^{-i2\pi(\frac{mu}{M}+\frac{nv}{N})}$$

cu coordonatele imaginii $u=0\ldots M-1$ şi $v=0\ldots N-1.$

Funcţii de bază 2D

În mod asemănător cu funcţiile de bază ale transformării monodimensionale, vom rescrie reprezentarea Euler a funcţiei ca sumă ponderată a funcţiilor bidimensionale complexe după cum urmează

$$e^{i2\pi(\frac{um}{M}+\frac{vn}{N})}=\underbrace{\cos[2\pi(\frac{um}{M}+\frac{vm}{N})]}_{C_{m,n}^{M,N}(u,v)}+i\cdot\underbrace{\sin[2\pi(\frac{um}{M}\frac{vn}{N})]}_{S_{m,n}^{M,N}(u,v)}$$

unde funcţiile $C_{m,n}^{M,N}$ şi $S_{m,n}^{M,N}$ reprezintă funcţiile cosinus şi sinus bidimensionale cu numărul de unde orizontale m şi cel de unde verticale n

$$C_{m,n}^{M,N}(u,v)=\cos[2\pi(\frac{um}{M}+\frac{vn}{N})]$$
$$S_{m,n}^{M,N}(u,v)=\sin[2\pi(\frac{um}{M}+\frac{vn}{N})]$$

Reprezentarea grafică în 2D

Reprezentarea grafică a funcţiilor complexe este în general dificilă. O variantă posibilă ar fi cea de reprezentanre a intensităţilor valorilor reale şi imaginare sub formă de suprafaţă grafică. Cea mai răspândită variantă este însă cea a reprezentării valorii absolute a spectrului complex $|G(m,n)|$.

Domeniul de valori

Majoritatea imaginilor naturale concentrează energia spectrală în zone de frecvenţe joase, cu valori maxime în zona originii. Din acest motiv, pentru a putea vizualiza şi frecvenţele joase de la periferie, se recurge la reprezentarea radicalului valorii absolute a spectrului $(\sqrt{|G(m,n)|})$ sau a logaritmului acesteia $(lg(|G(m,n)|))$.

Reprezentarea centrată

Dat fiind faptul că spectrul transformării este o funcție periodică, este indicată reprezentarea centrată a acestuia, deci având centrul axelor de coordonate în mijlocul imaginii, și alegând pentru coordonatele centrului axelor $(0,0)$ valorile (m,n) ale domeniului. Astfel vom avea:

$$|G(m,n)|=|G(-m,-n)|, unde -\frac{M}{2}\le m\le\frac{(M-1)}{2,} -\frac{N}{2}\le n\le\frac{(N-1)}{2}$$

Frecvențe și orientări în 2D

Prin descompunerea în funcțiile de bază sinus și cosinus, vom obține pentru spectru unde monodimensionale de-a lungul unei axe principale. Frecvența acestei unde fiind notată cu $\hat{f}=\frac{1}{\tau}$, având direcția dată de unghiul Ψ definită și fiind totodată efectivă.

În imaginea spectrală însă, poziția coordonatelor m și n este dată de

$$(m,n)=\pm\hat{f}\cdot(M\cos\Psi, N\sin\Psi)$$

și deci poziția coordonatelor spectrale nu corespunde, în mod normal, direcției imaginii.

Schematic, acest lucru este prezentat în Fig. 154.

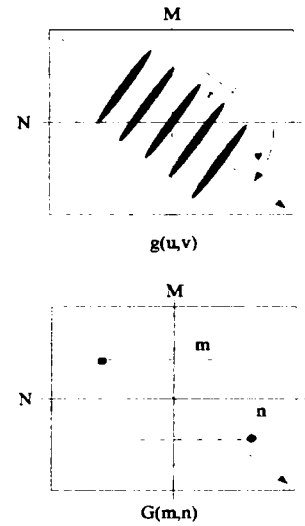


Fig. 154: Frecvența și orientarea spectrului 2D

Frecvența efectivă

Dat fiind faptul că numerele de undă m și n ne precizează câte perioade complete există în funcția de bază corespunzătoare de-a lungul a M și N unități pe direcție orizontală și respectiv verticală, iar concomitent considerând că frecvențele τ_x și τ_y de sampling sunt egale, frecvența efectivă poate fi calculată în modul următor:

$$\hat{f}(m,n)=\frac{1}{\tau}\sqrt{(\frac{m}{M})^2+(\frac{n}{N})^2}$$

Astfel, frecvența maximă de-a lungul axelor este

$$\hat{f}(\frac{\pm M}{2,0})=\hat{f}(0,\pm\frac{N}{2})=\frac{1}{\tau}\sqrt{(\frac{1}{2})^2}=\frac{1}{2\tau}=\frac{1}{2}f_s$$

Așadar, frecvența colțurilor spectrului va fi dată de

$$\hat{f}(\frac{\pm M}{2,}\pm\frac{N}{2})=\frac{1}{\tau}\sqrt{(\frac{1}{2})^2+(\frac{1}{2})^2}=\frac{1}{\sqrt{2}\tau}=\frac{\sqrt{2}}{2}f_s$$

și este deci de $\sqrt{2}$ ori mai mare decât cea de-a lungul axelor de coordonate.

Limite de frecvenţă şi aliasing

Considerând faptul că frecvenţa de sampling minimă este de-a lungul axelor de coordonate ale rasterului şi în scopul evitării fenomenului de aliasing, este necesar ca frecvenţa semnalului efectiv al imaginii de transformat să fie limitată la $\frac{f_s}{2}$ (în condiţiile tastării cu aceeaşi frecvenţă pe amble direcţii).

Orientarea

Direcţia unei unde sinusoidale bidimensionale cu coordonatele spectrale m şi n este dată de relaţia

$$\Psi(m,n)=\arctan_2(\frac{n}{N},\frac{m}{M})=\arctan_2(nM,mN),0\le m<M,0\le n<N$$

în punctul (0,0) fiind evident nedefinită.

De asemenea, un sinusoid bidimensional cu frecvenţa efectivă $\hat{f}$ şi direcţia Ψ va fi definit prin coordonatele sale spectrale

$$(m,n)=\pm\hat{f}(M\cos\Psi,N\sin\Psi)$$

Corecţia geometrică a spectrului

După cum se poate observa, în cazul unei unde sinusoidale cu direcţie de 45°, coeficienţii spectrali vor fi

$$(m,n)=\pm(\lambda M,\lambda N)\quad pentru-\frac{1}{2}\le\lambda\le\frac{1}{2}$$

şi deci se vor afla pe diagonala spectrului. În cazul în care însă spectrul nu este pătrat, deci $M\neq N$, direcţiile nu vor mai avea o înclinare de 45°.

Ca rezultat, o rotaţie cu un unghi α a imaginii (semnalului) va duce la o modificare a direcţiei undei spectrale în aceeaşi direcţie, dar cu un unghi diferit de α.

Transformarea rapidă Fourier (FFT)

Cooley şi Tukey au dezvoltat încă din 1965 un algoritm de implementare a transformărilor Fourier extrem de rapid [24].

FFT nu este altceva decât un algoritm de calcul a transformarilor discrete Fourier, mult mai rapid decât cele tradiţionale, a cărui explicaţie, preluată din [25], o vom prezenta în continuare.

Considerând transformarea discretă

$$S_d(n)=\sum_{k=0}^{N-1}b_{d0}(k)e^{-j\frac{2\pi nk}{N}},n=0,\ldots,N-1$$

unde am notat pentru simplificare $k=k\Delta x$ şi $n=n\Delta x$. Considerând că relaţia de mai sus conţine N ecuaţii, vom exemplifica funcţionarea algoritmului pentru $N=4$.

Utilizând substituţia

$$W = e^{-j\frac{2\pi}{N}}$$

vom obţine sistemul de ecuaţii liniare

$$S_d(n) = \sum_{k=0}^{N-1} b_{d0}(k) W^{kn}, n = 0, \ldots, N-1$$

care pentru cazul exemplului pot fi scrise şi sub forma

$$S_d(0) = b_{d0}(0)W^0 + b_{d0}(1)W^0 + b_{d0}(2)W^0 + b_{d0}(3)W^0$$
$$S_d(1) = b_{d0}(0)W^0 + b_{d0}(1)W^1 + b_{d0}(2)W^2 + b_{d0}(3)W^3$$
$$S_d(2) = b_{d0}(0)W^0 + b_{d0}(1)W^2 + b_{d0}(2)W^4 + b_{d0}(3)W^6$$
$$S_d(3) = b_{d0}(0)W^0 + b_{d0}(1)W^3 + b_{d0}(2)W^6 + b_{d0}(3)W^9$$

sau mai simplu, sub formă matricială extinsă

$$\begin{bmatrix} S_d(0) \\ S_d(1) \\ S_d(2) \\ S_d(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^4 & W^6 \\ 1 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} b_{d0}(0) \\ b_{d0}(1) \\ b_{d0}(2) \\ b_{d0}(3) \end{bmatrix}$$

sau compactă

$$S_d = W \cdot b_{d0}$$

Ţinând cont de faptul că W este o matrice complexă, iar b_{d0} eventual poate fi şi ea complexă, vom avea N^2 multiplicări complexe şi $N(N-1)$ adunări complexe de efectuat în vederea rezolvării operaţiei matriciale. Viteza ridicată de calcul a algoritmului FFT se datorează reducerii numărului acestor operaţii.

Numărul valorilor de sampling va fi fixat la o valoare $N = 2^i, i \in \mathbf{N}^+$. Astfel, pentru exemplul nostru, vom avea $i = 2$ şi $N = 4 = 2^2$.

Observând însă că între elementele matricii W există relaţia

$$W^{nk} = W^{nk \bmod N}$$

ca de exemplu în

$$W^9 = e^{\left(\frac{(-j2\pi)}{4}\right)9} = e^{-j\frac{9}{2}\pi} = e^{-j\frac{1}{2}\pi} = e^{-j\frac{2}{4}\pi} = e^{\frac{(-j2\pi)}{4}} = e^{\left(\frac{(-j2\pi)}{4}\right)1} = W^1$$

vom putea scrie

$$\begin{bmatrix} S_d(0) \\ S_d(1) \\ S_d(2) \\ S_d(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & 1 & W^2 \\ 1 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} b_{d0}(0) \\ b_{d0}(1) \\ b_{d0}(2) \\ b_{d0}(3) \end{bmatrix}$$

Matricea pătrată poate fi factorizată (demonstrația se găsește în [26]) sub forma

$$\begin{bmatrix} S_d(0) \\ S_d(2) \\ S_d(1) \\ S_d(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} b_{d0}(0) \\ b_{d0}(1) \\ b_{d0}(2) \\ b_{d0}(3) \end{bmatrix}$$

De remarcat schimbarea ordinii elementelor de la mijlocul vectorului S_d. Ecuația de mai sus se poate scrie și sub forma a două produse matriciale

$$\begin{bmatrix} b_{d1}(0) \\ b_{d1}(1) \\ b_{d1}(2) \\ b_{d1}(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} b_{d0}(0) \\ b_{d0}(1) \\ b_{d0}(2) \\ b_{d0}(3) \end{bmatrix}$$

și

$$\begin{bmatrix} S_d(0) \\ S_d(2) \\ S_d(1) \\ S_d(3) \end{bmatrix} = \begin{bmatrix} b_{d2}(0) \\ b_{d2}(1) \\ b_{d2}(2) \\ b_{d2}(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} b_{d1}(0) \\ b_{d1}(1) \\ b_{d1}(2) \\ b_{d1}(3) \end{bmatrix}$$

Astfel, elementul $b_{d1}(0)$ este rezultatul unei multiplicări și a unei adunări complexe

$$b_{d1}(0) = b_{d0}(0) + W^0 \, b_{d0}(2)$$

În mod identic se va obține $b_{d1}(1)$. În schimb, pentru calculul lui $b_{d1}(2)$ și $b_{d1}(3)$ nu mai avem nevoie decât de o adunare complexă, datorită faptului că $W^0 = -W^2$ iar în

$$b_{d1}(2) = b_{d0}(0) + W^2 \, b_{d0}(2) = b_{d0}(0) - W^0 \, b_{d0}(2)$$

produsul complex a fost deja calculat.

Așadar, vectorul complex b_{d1} necesită pentru calcul doar două produse complexe și patru adunări complexe.

Se poate demonstra astfel, că pentru $N = 2^i$, din factorizarea matricii $N \times N$ în i matrici de dimensiune $N \times N$ vom obține un raport de operații necesare de forma

$$\frac{t_{DFT}}{t_{FFT}} = \frac{N^2}{N\frac{i}{2}} = \frac{2N}{i}$$

ceea ce pentru un număr de tastări (samplings) de 1024 (2^{10}) conduce la o reducere a numului de operaţii şi deci a timpului de calcul cu un factor de circa $\frac{1}{200}$.

Transformarea discretă cosinus (DCT)

Transformările Fourier sunt în general utilizate pentru prelucrarea semnalelor complexe, generând un spectru complex chiar dacă semnalul original este compus în întregime din valori reale. Pentru reprezentarea semnalului nici partea reală singură şi nici cea imaginară nu sunt însă suficiente deoarece funcţiile corespondente sinus şi cosinus nu constituie prin sine un sistem de bază.

Pe de altă parte însă, ştim că un semnal real conduce la generarea unui spectru simetric, fapt pentru care partea complexă a rezultatului este redundantă, fiind necesar doar calculul unei jumătăţi a valorilor spectrale, fără a genera în acest mod pierderi de informaţie.

Din acest motiv, au fost implementate metode şi algoritmi de transformări spectrale asemănătoare DFT, dar care lucrează doar în spaţiul real. Dintre acestea, cel mai cunoscut sistem este Transformarea Discretă Cosinus (Discrete Cosinus Transformation DCT), utilizată în special în prelucrarea şi comprimarea imaginilor video [27].

Monodimensională

Forma generală a transformării este dată de relaţia următoare

$$G(m) = \sqrt{\frac{2}{M}} \sum_{u=0}^{M-1} g(u) \cdot c_m \cos\left(\pi \frac{m(2u+1)}{2M}\right), \qquad pentru\ 0 \le m < M$$

$$g(u) = \sqrt{\frac{2}{M}} \sum_{m=0}^{M-1} G(m) \cdot c_m \cos\left(\pi \frac{m(2u+1)}{2M}\right), \qquad pentru\ 0 \le u < M$$

unde

$$c_m = \begin{cases} \frac{1}{\sqrt{2}} & pentru\ m = 0 \\ 1 & altfel \end{cases}$$

De observat este faptul că variabilele u şi m sunt utilizate în mod diferit între transformarea directă şi cea inversă, şi prin urmare cele două transformări nu sunt identice.

Bidimensională

Formulele bidimensionale sunt obținute pornind de la cele monodimensionale, și deci

$$G(m,n)=\frac{2}{\sqrt{MN}}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} g(u,v)\cdot c_m\cos(\pi\frac{m(2u+1)}{2M})\cdot c_n\cos(\pi\frac{n(2v+1)}{2N})$$

$$=\frac{2\ c_m c_n}{\sqrt{MN}}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} g(u,v)\cdot D_m^M(u)\,D_n^N(v)$$

$$g(u,v)=\frac{2}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} G(m,n)\cdot c_m\cos(\pi\frac{m(2u+1)}{2M})\cdot c_n\cos(\pi\frac{n(2v+1)}{2N})$$

$$=\frac{2\ c_m c_n}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} G(m,n)\cdot D_m^M(u)\,D_n^N(v)$$

pentru $0\le m,u<M,\ 0\le n,v<N$ și unde am notat

$$D_m^M=\cos[\pi\frac{m(2u+1)}{2M}]=\cos[2\pi\frac{m(u+0.5)}{2M}]$$

Se observă deci, că perioada funcției de bază în acest caz este dublă față de DFT ($\frac{2M}{m}$ comparativ cu $\frac{M}{m}$), iar fazele funcțiilor sunt cu 0.5 unități deplasate.

5.2.3 Segmentarea

Prin segmentare se urmărește separarea obiectelor diferite existente în cadrul imaginii de restul conținutului, a obiectelor alipite și/sau a unui obiect în componentele sale.

Această etapă este asemănătoare cu cea de clasificare, doar că are loc la nivel de pixel, iar acum singura clasificare care are loc este împărțire pixelilor în apartenenți sau neapartenenți la obiect.

5.2.3.1 Morfologii ale imaginilor binare

După cum am mai amintit, imaginile binare sunt pentru prototiparea rapidă tipul de imagini cel mai potrivit, datorită avantajelor sale.

Prin faptul că algoritmii utilizați în continuare modifică conținutul informativ al imaginii, ei au un caracter morfologic, motiv pentru care modelele matematice utilizate poartă numele de morfologii matematice.

Elementul structural de bază

Orice imagine binară este compusă din elemente structurale de bază de forma

$$H(i,j)\in\{0,1\}$$

sau grafic ca în exemplul următor

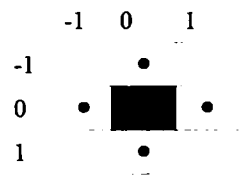


Fig. 155: Elementul structural de bază

Aceste elemente au sistemul lor propriu de coordonate, originea acestuia numindu-se hotspot (în figura de mai sus află în centru).

Mulțimi de puncte

În vederea definirii funcțiilor filtrelor morfologice este necesară definirea imaginii binare ca mulțime de coordonate ale punctelor. Pentru o imagine binară $I(u,v)$ această mulțime este dată de totalitatea coordonatelor pixelilor de valoare 1. Matematic, putem formula acest lucru sub forma

$$P_I=\{(u,v) \mid I(u,v)=1\}, u\in[0,\ M-1], v\in[0,\ N-1]$$

iar pentru elementul de bază

$$P_H=\{(u,v)\vee H(u,v)=1\}, u\in[-\frac{M_H}{2},\frac{M_H}{2}], v\in[-\frac{N_H}{2},\frac{N_H}{2}]$$

Valoarea mulțimii de puncte ale imaginii din Fig. 156 este

$$P_I=\{(1,0),(0,1),(2,1),(1,2),(4,4)\}$$

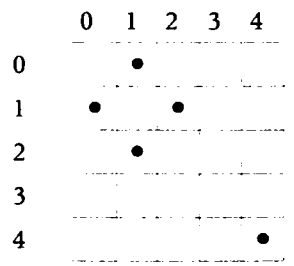


Fig. 156: Mulțimi de puncte

iar pentru elementul structural de bază dat ca exemplu în Fig. 155 este

$$P_H=\{(0,-1),(-1,0),(0,0),(1,0),(0,1)\}$$

dat fiind faptul că H are, după cum am amintit, propriul său sistem de axe de coordonate.

Operații binare

Grupul de operații binare sunt cele cunoscute din algebra binară (booleană), și fără de care nici una dintre operațiile următoare n-ar putea avea loc. Pentru o definire și notație unitară le vom aminti pe scurt în continuare.

Negația

Reprezintă trecerea unui pixel de la valoarea de background (de obicei 0) la valoarea de foreground (de

obicei 1) şi invers. Corespunde operaţiei unare NOT.

$$P_{\neg I} = \overline{P_I}$$

Reuniunea

Corespondenta operaţie binare OR. În urma ei toţi pixelii de valoare 1 din ambele imagini vor fi preluaţi. Prescurtat această operaţie o vom scrie sub forma

$$P_{I_1 \vee I_2} = P_{I_1} \cup P_{I_2}$$

Intersecţia

În urma acestei operaţii numai pixelii de valoare 1 din ambele imagini având aceleaşi coordonate vor prelua valoarea 1. Corespondenta operaţiei binare AND.

$$P_{I_1 \wedge I_2} = P_{I_1} \cap P_{I_2}$$

Deplasarea (shift)

Corespondenta operaţiilor binare de SHIFT (<< şi >>). Prin această operaţie pixelii vor fi mutaţi dintr-o zonă a imaginii în alta cu un număr de pixeli (i, j) . Deşi în algebra booleană există două tipuri de shift, noi vom comprima definiţia într-una singură, dependentă de semnele lui i şi j.

$$P_I \gg (i, j) = (x, y) = (x - i, y - j) \mid (i, j) \in [0 \ldots M-1, 0 \ldots N-1]$$

Diferenţa

Prin aplicarea acestei operaţii asupra două imagini binare, doar valorile de 1 ale primei imagini vor fi preluate care nu aparţin şi celei de-a doua imagini.

$$P_{I_1 \setminus I_2} = P_{I_1} \setminus P_{I_2}$$

Operaţii de bază

Pornind de la operaţiile de mai sus, în vederea prelucrării imaginilor binare, vom defini alte două operaţii foarte des folosite.

Dilataţia

Dilataţia este acea operaţie morfologică corespunzătoare termenului intuitiv de creştere, notată matematic în modul următor

$$I \oplus H = \{(x, y) = (u + i, v + j) \mid (u, v) \in P_I, (i, j) \in P_H\}$$

Din dilatatie rezultă aşadar toate combinaţiile posibile de puncte ale mulţimilor P_I şi P_H. Un exemplu simplu de dilataţie este prezentat în figura următoare.

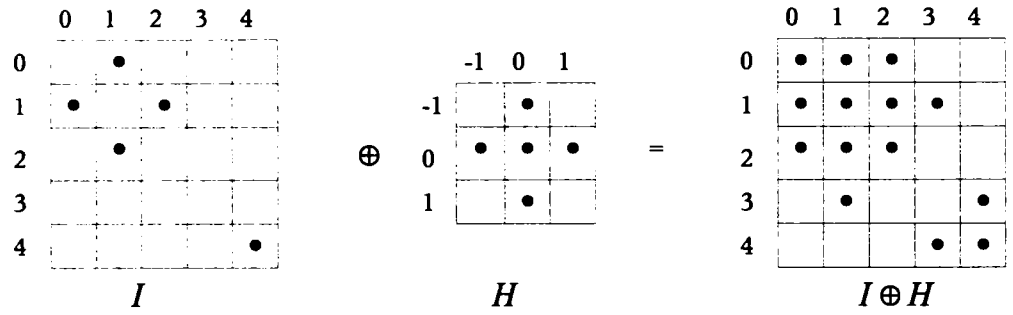


Fig. 157: Dilatația

Eroziunea

Eroziunea este operația inversă dilatației, definită prin

$$I \ominus H = \{(x,y) | (x+i, y+j) \in P_I, \forall (i,j) \in P_H\}$$

Cu alte cuvinte, eroziunea setează pixelul din hotspot doar în cazul în care fiecare dintre punctele cu valoare de 1 ale imaginii și cele ale structurii elementare coincid. O figură explicativă este prezentată în continuare.

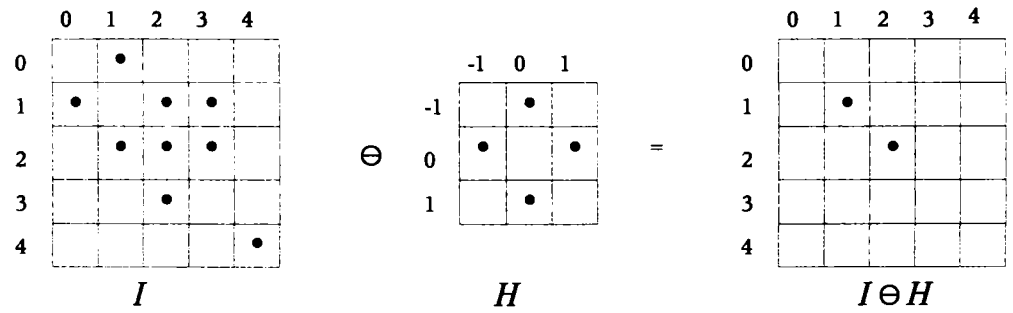


Fig. 158: Eroziunea

Proprietățile dilatației și eroziunii

Deși similare operațiilor cunoscute din teoria mulțimilor, aceste două operații nu sunt complementare – deși principial ar părea să fie. Înțelegerea proprietăților lor este însă esențială în vederea prelucrării imaginilor. O parte dintre cele mai importante proprietăți o vom prezenta în continuare, o amplă dezvoltare a acestui domeniu fiind expusă în [28].

Fiind date două structuri P_{I_1} și P_{I_2} atunci vom avea:

$$P_{I_1} \subseteq P_{I_2} \rightarrow I \ominus P_{I_1} \supseteq I \ominus P_{I_2} \rightarrow I \oplus P_{I_1} \subseteq I \oplus P_{I_2}$$

Dilatația este comutativă:

$$P_{I_1} \oplus P_{I_2} = P_{I_2} \oplus P_{I_1}$$

Eroziunea nu este comutativă:

$$P_{I_1} \ominus P_{I_2} \neq P_{I_2} \ominus P_{I_1}$$

Dilatația imaginii complementare este egală cu imaginea complementară a imaginii erodate:

$$\overline{I} \oplus P_I = \overline{I \ominus P_I}$$

Imaginea dilatată conține imaginea inițială iar în imaginea inițială este conținută imaginea erodată:

$$I \ominus P \subset I \subset I \oplus P$$

Eroziunea și dilatarea sunt transformări monoton descrescătoare sau crescătoare:

$$P_{I_1} \supseteq P_{I_2} \rightarrow P_{I_1} \oplus I \supseteq P_{I_2} \oplus I$$

$$P_{I_1} \supseteq P_{I_2} \rightarrow P_{I_1} \ominus I \supseteq P_{I_2} \ominus I$$

$$I \oplus (P_{I_1} \cup P_{I_2}) = (I \oplus P_{I_1}) \cup (I \oplus P_{I_2})$$

$$I \ominus (P_{I_1} \cup P_{I_2}) = (I \ominus P_{I_1}) \cap (I \ominus P_{I_2})$$

Prin combinarea eroziunii și dilatării obținem, analog adunării și scăderii:

$$(P_{I_1} \ominus P_{I_2}) \ominus P_{I_3} = P_{I_1} \ominus (P_{I_2} \oplus P_{I_3})$$

Morfologii complexe – filtre morfologice

Filtrele morfologice sunt definite prin două elemente. Tipul operației de efectuat și elementul de structură - a cărui mărime și formă sunt determinate de filtrul în care este utilizat.

Spre deosebire de filtrele liniare, nu este posibilă generarea unui element de structură H° isotropic, dat fiind faptul că elementele sale componente H_x și H_y vor genera totdeauna rezultate dreptunghiulare, deci neizotrope.

Filtrele complexe sunt în general obținute prin suprapunerea consecutivă a unor filtre morfologice simple, ca de exemplu în cazul "outline", deci de extragere a marginilor imaginii binare (filtru utilizabil în cazul prelucrării imaginilor scanate).

Ținând cont de faptul că marginile imaginii sunt determinate de pixelii aflați pe zona de frontieră între foreground și background (imaginea de front și fundal), vom genera în primul rând un filtru de eroziune, prin care imaginea de front va fi subțiată, iar negativul acestui rezultat îl vom intersecta cu imaginea inițială.

Ecuația filtrului este deci dată de

$$P = I \cap \overline{I \ominus H_n}$$

Importantă este observația că în cazul utilizării pentru H_n unei vecinătăți de 8 puncte vom obține rezultatul în vecinătăți de 4 puncte, și invers.

Deschiderea și închiderea imaginilor binare

Două cazuri speciale de operații binare sunt cunoscute și prezentate în literatura de specialite, și care datorită importanței lor, au primit nume semificative, ambele bazându-se pe eroziune și dilatație.

Deschiderea (Opening)

Deschiderea este operaţia compusă dintr-o eroziune urmată de o dilataţie cu acelaşi element de structură H_n.

Matematic, această operaţie poate fi descrisă sub forma:

$$I \circ H_n = (I \ominus H_n) \oplus H_n$$

Prin această operaţie se obţine eliminarea din imagine a tuturor elementelor mai mici decât elementul de structură utilizat, dat fiind faptul că în urma eroziunii acestea vor dispărea din imagine, iar în urma dilatării, restul formelor vor reveni la dimensiunea originală.

Închiderea (Closing)

Inversând ordinea operaţiilor din Opening, deci ca urmare a unei operaţii de dilatare urmată de una de eroziune cu acelaşi element structural, vom obţine operaţia de închidere (Closing), a cărei formulă matematică este:

$$I \bullet H_n = (I \oplus H_n) \ominus H_n$$

În urma acestei operaţii, interspaţii de dimensiuni mai mici decât a elementului de structură utilizat vor fi eliminate. Evident, în urma operaţiei de dilatare, aceste goluri vor fi umplute, iar în urma operaţiei de eroziune nu vor mai putea fi deschise, ele ne mai existând.

Proprietăţi ale operaţiilor de Opening şi Closing

Aceste două operaţii au şi două proprietăţi semificative – idempotenţa şi dualitatea.

Idempotenţa

Ambele operaţii sunt definite ca idempotente, deci rezultatul lor este finalizat în sensul că indiferent de numărul operaţiilor consecutive de acest tip asupra imaginii rezultatul operaţiilor rămâne neschimbat. Pe scurt, proprietarea de idempotenţă este descrisă de formulele:

$$(I \circ H_n) \circ H_n = I \circ H_n$$
$$(I \bullet H_n) \bullet H_n = I \bullet H_n$$

Dualitatea

Aceste două operaţii sunt denumite duale, datorită faptului că o deschidere a foreground-ului este identică cu închiderea background-ului şi invers.

$$I \circ H_n = \overline{(\bar{I} \bullet H)}$$
$$I \bullet H_n = \overline{(\bar{I} \circ H)}$$

Prelucrarea obiectelor individuale din imaginile binare

Toate operațiile pe care le-am descris anterior sunt efectuate asupra imaginilor binare în totalitate. De cele mai multe ori însă, apare necesitatea extragerii anumitor obiecte din cadrul acestei imagini, separarea lor de restul imaginii (fond sau alte obiecte). Din acest motiv, este evident de importanță majoră, punerea în evidență cel puțin a modalităților și posibilităților de extragere a obiectelor din imagine, operație numită "Separare", operație de multe ori necesară aplicării ulterioare a altor filtre și operații, din simplul motiv că ea elimină elementele neconcludente ale imaginii de analizat, restul operațiilor fiind efectuate doar asupra elementelor de importanță majoră.

"Grassfire"

Acest algoritm se bazează, așa cum îi spune și numele, pe modul de funcționare al incendiilor spontane în savane. Pornind de la unul sau mai multe puncte de plecare a focului, acesta se va extinde doar de-a lungul zonelor cu iarbă uscată.

În cazul prelucrării imaginii, vom utiliza o imagine de marcaj, ale cărei puncte determină locurile de pornire. În general această imagine poate conține unul sau mai multe puncte caracteristice, sau poate fi în mod automat generată printr-o operație de eroziune a imaginii inițiale. O altă aplicație foarte importantă este eliminarea obiectelor incomplete, deci a celor tangente la marginea imaginii, caz în care nu putem afirma cu certitudine că sunt complete. În acest caz, generarea unei măști cu un singur pixel ca ramă este elegantă, simplă și suficientă.

„Ultima Eroziune"

De multe ori însă, este necesar ca obiectele existentă să nu dispară comple în urma eroziunii, ci să păstreze o ultimă versiune a existenței lor înainte de dispariție. Din acest motiv, a fost implementat algoritmul de "Ultimă eroziune" (Ultimate erosion), a cărei diagrame de stare este prezentată în Fig. 159.

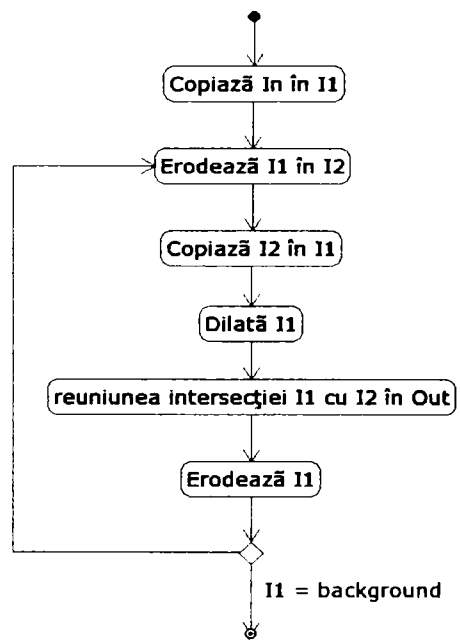


Fig. 159: Ultima eroziune

Transformarea „Totul sau nimic" (hit-or-miss)

În vederea recunoașterii elementelor de imagine mască în interiorul unei imagini date, este utilizată transformarea "hit-or-miss", în care elementul de structură este compus din două matrici – prima trebuind să existe în imaginea dată iar cea de-a doua în negativul acesteia. Valoarea punctelor din elementul structural $H=(H_1, H_2)$ poate fi deci nu numai 1 ci și 0. Pe larg, această transformare este redată în [29].

Matematic, putem reprezenta această transformare sub forma:

$$P = I \star H = (I \ominus H_1) \cap \overline{(I \oplus H_2)} = (I \ominus H_1) \cap (\overline{I} \ominus H_2)$$

unde $H = H_1 \cup H_2$ și $H_1 \cap H_2 = \emptyset$.

Un exemplu de mască este prezentată în Fig. 160, în care, considerând punctul de hotspot în colțul din stânga sus (0,0), vom avea următoarele valori ale mulțimilor amintite:

$$H_1 = \{(1,1)\}$$
$$H_2 = \{(1,0),(0,1),(2,1),(1,2)\}$$

Pixelii notați cu x nu au nici o importanță pentru operație și vor fi ignorați.

x	0	x
0	1	0
x	0	x

Fig. 160: Elementul structural al transformării "hit-or-miss"

„Slăbirea"(thinning) și „îngrășarea" (thickening)

Pornind de la transformarea "totul sau nimic" sunt definite următoarele două operații.

Slăbirea este dată de relația

$$P = I \square H = I \setminus (I \star H) = I \cap \overline{(I \star H)}$$

unde cazul cel mai important, în care grosimea rezultată este de un pixel, este denumit "Scheletare" și va fi prezentat în continuare.

Scheletarea

După cum am văzut, scheletarea este un caz particular de "slăbire", care prezintă următoarele proprietăți:

- linia scheletului este de un singur pixel
- linia scheletului trebuie să fie aflată la mijlocul obiectului, deci la o distanță egală de marginile acestuia
- linia scheletului n-are voie să se mai modifice în urma operațiilor consecutive de slăbire

În general scheletarea are loc prin aplicarea consecutivă secvențială a transformării de subțiere prin utilizarea mai multor elemente structurale H_n. În [30] este prezentată o succesiune de astfel de elemente de tip 3×3 precum și o ordine preferențială de apelare a operațiilor. Acest exemplu îl vom prezenta în continuare succint.

1	x	0	x	0	0	0	0	0	x
1	1	0	1	1	0	x	1	x	0	1	1
x	x	0	x	1	x	1	1	x	x	1	x
H_0			H_1			H_2			H_3		
0	x	x	x	1	x	x	1	1	x	1	x
0	1	1	0	1	1	x	1	x	1	1	0
0	1	x	0	0	x	0	0	0	x	0	0
H_4			H_5			H_6			H_7		

Ordinea recomandată de apelare a operaţiilor de slăbire este:

$$I = I \square H_0$$
$$I = I \square H_4$$
$$I = I \square H_2$$
$$I = I \square H_2$$
$$I = I \square H_1$$
$$I = I \square H_5$$
$$I = I \square H_3$$
$$I = I \square H_7$$

Datorită faptului că transformarea necesită putere de calcul proporţională cu dimensiunea imaginii, este necesară implementarea de algoritmi optimizaţi sau sisteme hard aferente. De exemplu, pentru transformarea unei imagini de dimensiune 512×512 sunt necesare 28672 de operaţii de bază.

Îngroşarea

Îngroşarea este definită ca fiind reuniunea mulţimilor imaginii iniţiale cu cea a transformării "totul sau nimic" (hit-or-miss), şi este dată de relaţia

$$P = I \blacksquare H = I \cup (I \bigstar H)$$

5.2.3.2 Numărarea obiectelor după Euler

Ca aplicaţie a transformării "totul sau nimic" se poate arăta cât de simplă este numărarea obiectelor din cadrul imaginii utilizând formula lui Euler:

$$n - h = v + f - e$$

în care am notat cu:

n – numărul de obiecte (number)

h – numărul de goluri (holes)

v – numărul de colţuri (vertices)

f – numarul de faţete (faces)

e – numărul de canturi (edges)

Numărarea obiectelor în raster hexagonal

Din analiza vecinătăţilor punctelor din rasterul unei imagini binare hexagonale, se poate simplu determina numărul de obiecte ale imaginii. Astfel, elementele de mai sus, ignorând pentru început golurile care vor necesita un pas suplimentar (h=0), vor putea fi calculate în modul următor:

$$v = I\begin{pmatrix} \\ 1 \end{pmatrix}$$

$$f = I\begin{pmatrix} & 1 & \\ 1 & & 1 \end{pmatrix} + I\begin{pmatrix} 1 & & 1 \\ & 1 & \end{pmatrix}$$

$$e = I\begin{pmatrix} & \\ 1 & 1 \end{pmatrix} + I\begin{pmatrix} 1 & \\ & 1 \end{pmatrix} + I\begin{pmatrix} & 1 \\ 1 & \end{pmatrix}$$

unde valorile de 1 desemnează poziția pixelilor de valoare 1 din rasterul binar al imaginii.

Spre exemplificare, considerând exemplul din Fig. 161, vom avea

$$v = 9$$
$$f = 2$$
$$e = 7$$
$$h = 0$$

deci

$$n = 9 + 2 - 7 = 4$$

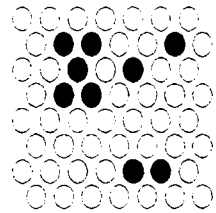


Fig. 161: Numărarea obiectelor în raster hexagonal

Ecuația de calcul poate fi scrisă și sub forma:

$$\underbrace{N_6}_{n} = \underbrace{I\begin{pmatrix} \\ 1 \end{pmatrix}}_{v} + \underbrace{I\begin{pmatrix} & 1 & \\ 1 & & 1 \end{pmatrix} + I\begin{pmatrix} 1 & & 1 \\ & 1 & \end{pmatrix}}_{f} - \underbrace{I\begin{pmatrix} & \\ 1 & 1 \end{pmatrix} - I\begin{pmatrix} 1 & \\ & 1 \end{pmatrix} - I\begin{pmatrix} & 1 \\ 1 & \end{pmatrix}}_{e}$$

$$N_6 = 9 + 1 + 1 - 3 - 2 - 2$$

Această ecuație se poate simplifica rezultând formula generală de calcul a numărului de obiecte într-un raster hexagonal

$$N_6 = I\begin{pmatrix} & 1 & \\ 0 & & 0 \end{pmatrix} - I\begin{pmatrix} 1 & & 1 \\ & 0 & \end{pmatrix}$$

Formula funcționează însă numai în cazul în care figura nu are goluri $(h = 0)$.

Numărarea în raster dreptunghiular

Numărarea în raster dreptunghiular este asemănătoare cu cea din rasterul hexagonal, trebuind însă să se facă deosebire între tipurile de vecinătăți considerate.

Astfel pentru o vecinătate de tip 4 vom avea formula de calcul dată de

$$N_4 = I\begin{vmatrix} 1 & 0 \\ 0 & x \end{vmatrix} - I\begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix}$$

iar pentru vecinătăți de tip 8

$$N_8 = I\begin{vmatrix} 1 & 0 \\ 0 & 0 \end{vmatrix} - I\begin{vmatrix} x & 1 \\ 1 & 0 \end{vmatrix}$$

Un exem lu de calcul entru o vecinătate de ti 8 este redat în figura alăturată.

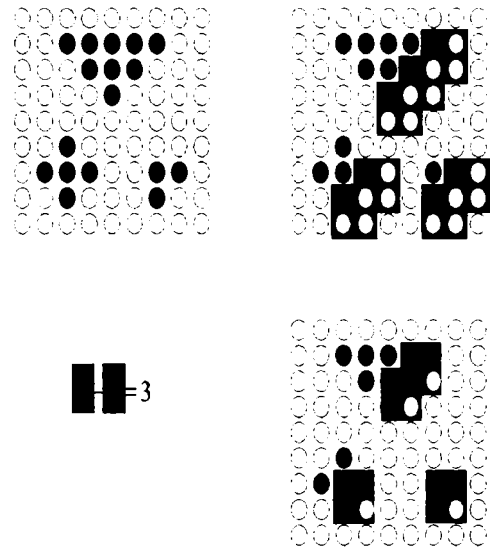


Fig. 162: Numărarea obiectelor în raster dreptunghiular

5.2.3.3 Transformări geometrice (afine)

Toate operațiile amintite până acum se referă doar la modificări care nu schimbă geometria imaginii. De multe ori, este însă necesară și modificarea acesteia, în cazul în care imaginile fiind reprezentări ale obiecte or 3 ex stente n rea tate, mens un e ș /sau proporțiile acestora trebuie modificate în vederea prelucrării lor.

În principiu, o transformare geometrică este acel tip de transformare prin care punctele imaginii de intrare își păstrează valorile, modificările având loc asupra poziției acestora, deci

$$I(u,v) \rightarrow I'(u',v')$$

Notând cu $i(u,v)$ coordonatele ale unui punct al imaginii inițiale $I(u,v)$ și cu $i'(u',v')$ cele ale punctului corespondent al imaginii de ieșire (transformate) I', forma generală a unei astfel de transformări este:

$$\boldsymbol{i}' = T(\boldsymbol{i}), \quad T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

De asemenea, descompunând funcția după cele două axe de coordonate existente, vom putea nota

$$\boldsymbol{i}'_u = T_u(u,v), \boldsymbol{i}'_v = T_v(u,v)$$

Transformările domeniului bidimensional prezintă trei operații principale, pe care le vom prezenta în continuare, prin intermediul cărora putem genera toate celelalte modificări imaginabile. Aceste operații poartă în algebra liniară denumirea de transformări afine.

Deși aceste operații sunt, după cum am amintit operații definite și cu valori în spațiul bidimensional, este mai convenabil să definim operațiile în spațiul tridimensional iar rezultatele să le proiectăm în 2D.

Din acest motiv, vom utiliza în continuare în vederea aplicării transformărilor afine matrice de tip 3×3 și vectori de tip 3×1 de forma

$$\boldsymbol{\tau} = \begin{pmatrix} \tau_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & \tau_{22} & \tau_{23} \\ \tau_{31} & \tau_{32} & \tau_{33} \end{pmatrix}$$

și

$$V = \begin{pmatrix} v_{11} \\ v_{12} \\ v_{13} \end{pmatrix}$$

Astfel, coordonatele punctelor imaginilor vor putea fi notate sub forma matricială

$$I=\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \text{ și respectiv } I'=\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix}$$

Ținând cont de faptul că

$$I'=\mathbf{T}\cdot I$$

sau

$$\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix}=\begin{pmatrix} \tau_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & \tau_{22} & \tau_{23} \\ \tau_{31} & \tau_{32} & \tau_{33} \end{pmatrix}\cdot\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

putem observa că aceast sistem nu poate fi rezolvat decât în cazul în care $\tau_{31}=\tau_{32}=0$ și $\tau_{33}=1.$

deci forma generală a matricii $\mathbf{T}$ va fi dată de

$$\mathbf{T}=\begin{pmatrix} \tau_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & \tau_{22} & \tau_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

Translația

Translația este modificarea poziției punctului de intrare după un vector de deplasare $\mathbf{d}(d_u, d_v)$

deci

$$\boldsymbol{T}=\begin{pmatrix} 1 & 0 & d_u \\ 0 & 1 & d_v \\ 0 & 0 & 1 \end{pmatrix}$$

Scalarea

Scalarea este prin definiție modificarea dimensiunii imaginii inițiale după cele două axe de coordonate, obținută prin dilatarea sau comprimarea lor prin intermediul vectorului $\mathbf{s}(s_u, s_v)$

$$\boldsymbol{S}=\begin{pmatrix} s_u & 0 & 0 \\ 0 & s_v & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotația

Rotație este operația prin care coordonatele unui punct al imaginii inițiale este rotit în jurul originii axelor de coordonate cu un unghi dat α.

$$\boldsymbol{R}=\begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Transformarea completă

O transformare completă este rezultatul aplicării succesive a transformărilor afine. De exemplu, o rotație urmată de o translație va fi dată de

$$\boldsymbol{T}_{TR}=\boldsymbol{R}\cdot\boldsymbol{T}$$

iar cea inversă de

$$\boldsymbol{T}_{TR}^{-1}=\boldsymbol{T}^{-1}\cdot\boldsymbol{R}^{-1}$$

5.2.3.4 Extracția contururilor

Pentru ochiul uman, modificarea bruscă a valorilor intensității sau culorilor duce la noțiunile de contur, margine și colțuri. Pentru creierul uman, aceste informații reprezintă unele dintre cele mai importante elemente ale vizualizării, și chiar în cazul lipsei tuturor celorlalte informații, imaginea rezultată este interpretabilă. Dintr-o simplă schiță în alb și negru, creierul uman generează fără probleme corelația cu obiectul reprezentat.

În cadrul scanării în vederea prototipării rapide determinarea zonelor de margine, a contururilor și a colțurilor este mult prea importantă pentru a nu fi analizată în cadrul lucrării de față.

Detectarea contururilor prin gradienți

Dat fiind faptul că imaginile binare sau în nuanțe de gri pot fi definite ca funcții, avem posibilitatea utilizării derivatelor pentru domeniile continui ale acestora. Luând în considerare de exemplu carul unei imagini în nuanțe de gri, histograma acesteia de-a lungul unei linii are o formă funcțională $f(u)$, asemănătoare figurii următoare:

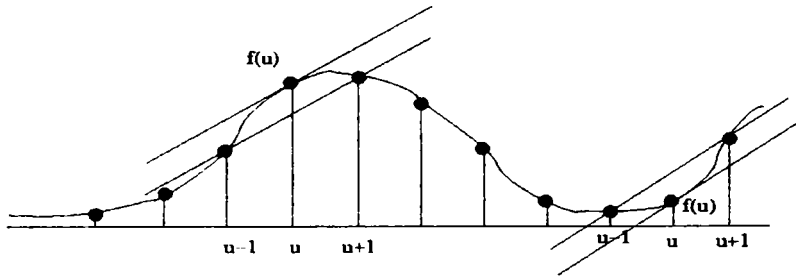


Fig. 163 Discretizarea primei derivate

Astfel, considerând valorile discrete ale funcţiei, putem determina valoarea aproximativă a primei derivate

$$f'(u)=\frac{df}{du}(u)\approx\frac{f(u+1)-f(u-1)}{2}$$

Din exemplul următor, al unei imagini de luminozitate variabilă, reiese reprezentarea grafică a primei derivate corespunzătoare.

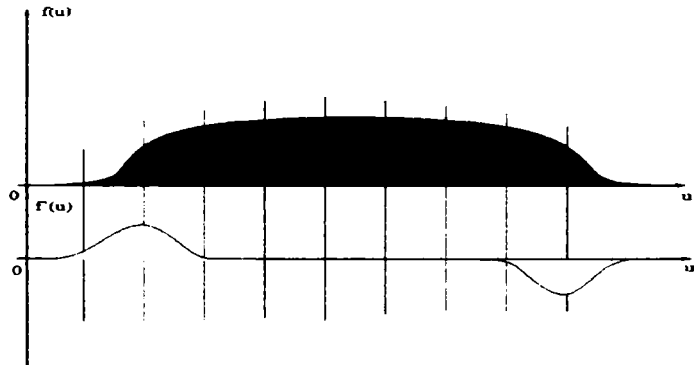


Fig. 164 Reprezentarea primei derivate

Derivarea discretă de ordinul unu

În cazul imaginilor, avem însă de-a face cu funcţii bidimensionale, astfel încât, derivarea trebuie făcută parţial, pentru fiecare dintre coordonaltele imaginii, aşadar $\frac{\partial I}{\partial u}(u,v)$ şi $\frac{\partial I}{\partial v}(u,v)$.

Vectorul dat de

$$\nabla I(u,v)=\begin{bmatrix}\frac{\partial I}{\partial u}(u,v)\\ \frac{\partial I}{\partial v}(u,v)\end{bmatrix}$$

poartă numele de vectorul gradienților sau pe scurt gradientul funcției I în punctul de coordonate (u, v). Modulul acestuia este dat de

$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial u}\right)^2 + \left(\frac{\partial I}{\partial v}\right)^2}$$

și este invariant cu rotirea imaginii, și deci independent de orientarea acesteia. Această proprietate este foarte importantă pentru localizarea muchiilor, fiind elemente de bază pentru numeroși operatori.

Filtre pentru detectarea muchiilor

Folosindu-ne de Fig. 164, putem defini un filtru liniar ca în formula următoare

$$H_u = \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} = 0.5 \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

pentru prima axă de coordonate, unde coeficientul de 0.5 este datorat faptului că punctul de maxim este, după cum reiese și din figură, la mijlocul intervalului dintre u și $u+1$. Pentru cea de-a doua coordonată, vom avea în mod asemănător

$$H_v = \begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix} = 0.5 \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

După cum se vede din figurile următoare, filtrul vertical este foarte sensibil la variații orizontale ale intensităților pe când cel orizontal reacționează cel mai puternic la variații bruște ale intensităților verticale. Modulul gradientului are după cum se poate cu ușurință observa, o variația cea mai puternică în zonele de contur.

Dintre cele mai cunoscute și interesante filtre, care lucrează pe acest principiu, vom prezenta câteva în continuare. Imaginea de referință utilizată pentru detectarea muchiilor este cea dată în Fig. 165:

Fig. 165: Detectarea muchiilor - figura de referință

Operatorul Prewitt

Filtrul bazat pe diferențiere 3×3

$$H_u^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ și } H_v^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Descompunând aceste matrici în produse vectoriale, obținem

$$H_u^P = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \text{ și } H_v^P = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

așadar, vom avea trei neteziri de tip "Box" atât peste linii cât și peste coloane, care, datorită proprietății de comutativitate a suprapunerii, permit aplicarea lor în orice ordine.

Valoarea gradientului este astfel dată de

$$\nabla I(u,v) \approx \frac{1}{6}\begin{bmatrix} H_u^P * I \\ H_v^P * I \end{bmatrix}$$

Un exemplu de detectare a muchiilor prin intermediu acestui operator este prezentată în imaginea următoare:

Fig. 166: Detectarea muchiilor - Prewitt

Operatorul Sobel

Este un operator asemănător celui aminti anterior, singura deosebire fiind reprezentată de valorile conținute de matricile filtrelor:

$$H_u^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ și } H_v^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Descompunând aceste matrici în produse vectoriale, obținem

$$H_u^S = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \text{ și } H_v^S = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

cu valoarea gradientului

$$\nabla I(u,v) \approx \frac{1}{8}\begin{bmatrix} H_u^S * I \\ H_v^S * I \end{bmatrix}$$

Exemplul următor utilizează acest operator în vederea detectării muchiilor.

Fig. 167: Detectarea muchiilor - Sobel

Operatorul Roberts

Unul dintre primii operatori utilizaţi în vederea detectării muchiilor, utilizează matrici de dimensiune 2×2 de forma

$$H_u^R = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \text{ şi } H_v^R = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

După cum ne putem imagina, fapt care reiese şi din figura următoare, filtrul reacţionează puternic la variaţiile diagonale ale intensităţilor punctelor imaginii de analizat.

Fig. 168: Detectarea muchiilor - Roberts

Operatorul Kirsch

În vederea alegerii unui filtru optimal, trebuie să luăm în considerare faptul că este necesar un compromis între calitatea obţinerii muchiilor şi unghiul sub care filtrul acţionează efectiv. Astfel, cu cât un filtru trebuie să reacţioneze mai precis asupra unei imagini, cu atât mai mic este unghiul sub care acest filtru este de efectivitate ridicată. Soluţiile generale sunt reprezentate prin utilizarea unei perechi de filtre relativ 'largi' pe două direcţii perpendiculare sau a mai multor filtre 'înguste' din mai multe direcţii.

Un astfel de filtru este prezentat de Kirsch ([31]), fiind constituit din 8 filtre ce acţionează în-un interval de 45°:

$$H_0^K=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} \quad H_1^K=\begin{bmatrix}-2 & -1 & 0\\ -1 & 0 & 1\\ 0 & 1 & 2\end{bmatrix}$$

$$H_2^K=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix} \quad H_3^K=\begin{bmatrix}0 & -1 & -2\\ 1 & 0 & -1\\ 2 & 1 & 0\end{bmatrix}$$

$$H_4^K=\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix} \quad H_5^K=\begin{bmatrix}2 & 1 & 0\\ 1 & 0 & -1\\ 0 & -1 & -2\end{bmatrix}$$

$$H_6^K=\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix} \quad H_7^K=\begin{bmatrix}0 & 1 & 2\\ -1 & 0 & 1\\ -2 & -1 & 0\end{bmatrix}$$

Din aceste opt filtre, trebuie însă calculate doar patru, celelalte fiind, după cum se vede din matricele anterioare, doar de semn schimbat. Putem deci scrie:

$$H_{(4+k)}^K=-H_k^K, \forall\, k\in[0,3]$$

şi datorită liniarităţii operaţiei de suprapunere

$$I*H_{(4+k)}^K=I*(-H_k^K)=-(I*H_k^K), \forall\, k\in[0,3]$$

Dimensiunea şi direcţia muchiilor

Indiferent de tipul operatorului utilizat, gradienţii după direcţiile de filtrare prezintă o anumită grosime a muchiilor determinate precum şi o direcţie a acestora.

În cazul utilizării unui operator cu filtre după două direcţii vom obţine două imagini filtrate $F_u(u,v)$ şi $F_v(u,v)$ date după cum am văzut de formulele

$$F_u(u,v)=I*H_u(u,v) \quad \text{şi} \quad F_v(u,v)=I*H_v(u,v)$$

Notând cu $E(u,v)$ grosimea muchiilor obţinute prin suprapunerea acestor imagini, aceasta va putea fi exprimată în modul următor:

$$E(u,v)=\sqrt{(F_u(u,v))^2+(F_v(u,v))^2}$$

iar direcţia locală (unghiul de înclinare) a acesteia este dată de

$$\Phi(u,v)=\arctan\left(F_u\frac{(u,v)}{F_v}(u,v)\right)$$

În cazul operatorului Kirsch vom avea

$$E^K(u,v)=max(F_i(u,v))=max(|F_j(u,v)|),\forall i\in[0,8],j\in[0,3]$$

respectiv

$$\Phi^K(u,v)=\frac{\pi}{4}j,\ j=\underset{0\le j\le 7}{argmax}(F_i(u,v))$$

Singurul avantaj al operatorului Kirsch este în practică faptul că nu utilizează radicali, şi deci calculul rezultatelor este mai rapid.

Derivata discretă de ordinul doi

Idea principală care se ascunde în spatele raţionamentelor legate de utilizarea derivatei de ordinul doi, rezidă din faptul că astfel este permisă o determinare mai exactă a punctului de maxim a primei derivate, dat fiind faptul că acesta este dat de punctele în care valoarea celei de-a doua derivate este zero. Aşadar în acest caz, punctele de zero ale celei de-a doua derivat sunt de fapt coordonatele muchiei căutate.

Din nefericire, cea de-a doua derivată este foarte sensibilă la bruiaje, fapt pentru care în prealabilul utilizării filtrelor bazate pe ea, este necesară o netezire a imaginii iniţiale.

Un alt impediment este dat de faptul că, de foarte multe ori, ceea ce ochiul uman determină în mod subiectiv ca fiind o muchie, diferă de ceea ce operatorii detectă, din cel puţin două motive:

- operatorii reacţionează doar la nivel local de variaţie a intensităţii, în timp ce sistemul uman vizual compensează aceste intensităţi sau chiar completează muchii întrerupte (chiar cu riscul apariţiei iluziilor optice)
- muchiile nu apar din cauza utilizării unei anumite rezoluţii sau a unei scări de scanare ci datorită mai multor nivele de scalare, ceea ce în două dimensiuni nu este posibil

Astfel, operatorii de detectare a muchiilor nu pot reacţiona decât în intervalul dat de rangul matricii utilizate de filtru. Din acest motiv, este necesară fie introducerea de operatori de dimensiuni mari (şi deci a filtrelor respective), fie micşorarea imaginii de analizat. Acest principiu stă la baza tehnicilor 'Multi-Rezoluţie' cum ar fi metodele ierarhice sau piramidale [32].

Filtrul Canny

Filtrul Canny este un exemplu de filtru bazat pe utilizarea filtrelor direcţionale de dimensiuni mari în mai multe rezoluţii ale imaginii ale căror rezultate sunt apoi îmbinate într-o aşa numită "edge map", deci o hartă a muchiilor. Prin această metode se urmăreşte atingerea concomitentă a trei scopuri:

- minimizarea numărului de muchii fals determinate
- poziţionarea cât mai exactă a muchilor
- raportarea a unei singure muchii determinate

Chiar dacă acest filtru este utilizat doar pentru o singură rezoluţie, prin adaptarea razei de acţiune a filtrului (σ) se obţine în general o detectare mai precisă decât prin utilizarea operatorilor simpli.

Filtrul LoG

Acest filtru utilizează şi el cea de-a doua derivată a funcţiei intensităţii imaginii, însă prin extragerea ponderată a valorii celei de-a doua derivate din funcţia iniţială. Datorită faptului că în zona muchiei cea de-a doua derivată schimbă semnul, prin adunarea acestor valori la funcţia dată se va obţine o claritate ridicată a poziţiei muchiei. Funcţia astfel determinată este dată deci de formula:

$$\hat{f}(u) = f(u) - w f''(u)$$

unde w reprezintă factorul de pondere amintit.

Dat fiind faptul că este utilizată cea de-a doua derivată, este necesară o netezire prealabilă a imaginii – prin utilizarea unui filtru Gauss.

În cazul imaginii, având de-a face cu funcţii bidimensionale, este necesară utilizarea operatorului Laplace pentru determinarea celei de-a doua derivate, deci:

$$(\nabla^2 f)(u,v) = \frac{\partial^2 f}{\partial^2 u}(u,v) + \frac{\partial^2 f}{\partial^2 v}(u,v)$$

În mod identic cu cel prezentat în cazul derivatei de ordinul unu, vom putea determina valorile discrete ale acestei funcţii prin utilizarea de filtre liniare simple cum ar fi:

$$\frac{\partial^2 f}{\partial^2 u} \approx H_u^L = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \text{ şi } \frac{\partial^2 f}{\partial^2 v} \approx H_v^L = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

care împreuna vor duce la filtrul bidimensional

$$H^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Se observă, că suma coeficienţilor utilizaţi este zero, ca de fapt în cazul tuturor filtrelor bazate pe gradienţi, fapt pentru care, numărul filtrelor de acest tip este nelimitat, atât timp cât este respectată această condiţie şi cea de simetrie. Alte două exemple de filtre LoG construite după acelaşi principiu pot fi:

$$H_{8sup}L = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ şi } H^L = \begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Fig. 169: Detectarea muchiilor - LoG

Rezultatul aplicării acestui filtru este prezentat în Fig. 169.

5.2.3.5 *Extracţia morfologică a contururilor*

Zamperoni a demonstrat în lucrarea sa [33] că metodele de extragere a contururilor pot fi aplicate şi asupra imaginilor gri, în cazul în care din imaginea dilatată se extrage imaginea erodată, aşadar:

$$P = (I \oplus H) - (I \ominus H)$$

În cazul în care există posibilitatea utilizării unei imagini nivelate, Lee, Haralik şi Shapiro propun în [34] următoarea formulă:

$$P = min\{(I - (I \ominus H)), ((I \oplus H) - I)\}$$

5.2.3.6 Descoperirea colţurilor

În special în vederea detectării poziţiei anumitor puncte în imagini succesive, un rol foarte important îl prezintă punctele semificative ale contururilor, puncte în care zone de contur se intersectează în cadrul imaginii digitale. Aceste puncte semificative le vom denumi colţuri sau puncte de interes (points of interest).

Puncte de interes (Points of Interest)

Problema principală a unui detector bun de puncte de interes este dată de faptul că acesta trebuie să fie în stare să detecteze aceste puncte într-un mod precis şi reproductibil, chiar în cazul bruiajelor imaginii.

În general, aceste detectoare se bazează pe faptul că o margine apare în momentul modificării intensităţii pixelilor imaginii, şi deci zone în care gradientul funcţiei imaginii într-un anumit punct este ridicat într-o anumită direcţie, iar pe direcţia perpendiculară este redus, într-o zonă a unui punct de interes acest gradient este ridicat pe două sau mai multe direcţii.

Detectorul Harris

Deşi la ora actuală există detectori de cante bazaţi pe algoritmi mai puternici, unul dintre detectorii de bază este cel pe care-l vom prezenta în continuare, detector implementat şi prezentat de Harris şi Stephens în [35].

Principiul de bază este bazat, aşa cum am amintit anterior pe faptul că gradientul funcţiei imaginii are valori ridicate pe cel puţin două direcţii, şi utilizează aşadar derivatele parţial ale funcţiei imaginii după cele două direcţii (orizontală şi verticală).

Matricea de structură locală

Derivatele parţiale ale funcţiei imaginii sunt definite de

$$I_u(u,v) = \frac{\partial I}{\partial u}(u,v) \quad \text{şi} \quad I_v(u,v) = \frac{\partial I}{\partial v}(u,v)$$

Pentru fiecare pixel al imaginii vor fi calculate trei valori:

$$A(u,v) = I_u^2(u,v)$$
$$B(u,v) = I_v^2(u,v)$$
$$C(u,v) = I_u(u,v) \cdot I_v(u,v)$$

cu ajutorul cărora va fi determinată matricea de structură locală $M(u,v)$ dată de formula următoare:

$$M(u,v) = \begin{pmatrix} I_u^2 & I_u \cdot I_v \\ I_u \cdot I_v & I_v^2 \end{pmatrix} = \begin{pmatrix} A & C \\ C & B \end{pmatrix}$$

Acestei matrici i se va aplica un filtru de netezire Gauss $H_{G,\sigma}$ astfel încât vom obține:

$$\overline{M}=\begin{pmatrix} A*H_{G_\sigma} & C*H_{G_\sigma} \\ C*H_{G_\sigma} & B*H_{G_\sigma} \end{pmatrix}=\begin{pmatrix} \overline{A} & \overline{C} \\ \overline{C} & \overline{B} \end{pmatrix}$$

Matricea de mai sus, datorită simetriei sale, o vom putea scrie sub forma:

$$\overline{M}=\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

în care am notat cu λ_1 şi λ_2 valorile proprii ale matricii $\overline{M}$.

$$\lambda_{1,2}=\frac{trace(\overline{M})}{2}\pm\sqrt{\left(\frac{trace(\overline{M})}{2}\right)^2-det(\overline{M})}$$
$$=\frac{1}{2}\left(\overline{A}+\overline{B}\pm\sqrt{\overline{A}^2-2\overline{A}\,\overline{B}+\overline{B}^2+2\overline{C}^2}\right)$$

În cazul unei imagini uniforme, vom avea $\overline{M}=0$ şi deci şi valorile proprii vor avea valoarea 0 respectiv $trace(\overline{M})=0$ funcţia trace fiind suma valorilor proprii al matricii. În cazul unei margini, indiferent de direcţia acesteia, vom avea $\lambda_1>0$ şi $\lambda_2=0$. Valorile proprii vor determina astfel calitatea marginii iar vectorii proprii corespunzători ne vor da direcţia acesteia. În cazul unui punct de interes, deci a unui colţ, ambele valori proprii vor fi diferite de 0.

Funcţia de răspuns a colţului

În vederea determinării colţurilor, vom considera valoarea diferenţei valorilor proprii ale matricii $\overline{M}$ care, în acest caz va trebui să fie cât mai mic. Astfel, detectorul Harris defineşte ca etalon de măsură pentru valoarea intensităţii colţului funcţia

$$Q(u,v)=det(\overline{M})-\alpha\cdot\left(trace(\overline{M})\right)^2=(\overline{A}\,\overline{B}-\overline{C}^2)-\alpha\cdot(\overline{A}+\overline{B})^2$$

în care α are o valoare fixă, în domeniul 0.04...0.06 (maximal 0.25). Sensibilitatea detectorului este invers proporţională cu α - valori mari vor fi detecta mai puţine colţuri.

Determinarea colţurilor

Pentru acceptarea unui punct ca fiind un colţ, este utilizată o treaptă de valoare ridicată, care comparată cu valoarea actuală a funcţiei de răspuns va avea valori inferioare acesteia, sau în exprimare matematică, notând cu t_H valoarea treptei amintite:

$$Q(u,v)>t_H$$

Valoarea lui t_H este în general aleasă în intervalul $[100.000\ldots1.000.000]$.

Punctele astfel determinate sunt înşirate într-un vector. Pentru a evita existenţa colţurilor foarte apropiate, ca urmare a erorilor de determinare sau a bruiajelor imaginii, punctele astfel rezultate vor fi sortate iar din punctele colţurilor geometric apropiate va fi ales doar cel de intensitate maximă.

5.2.3.7 Detectarea curbelor simple

Problema apartenenței punctelor de pe muchiile determinate ale imaginilor pare să fie o problemă trivială, din moment ce contururile au fost determinate. În realitate însă, această problemă este complicată, în special din cauza faptului că punctele de contur determinate aparțin unor construcții aproximate (chiar dacă foarte bine), iar existența lor este definită pe domenii mai mult sau mai puțin înguste. Pe de altă parte, tocmai această constatare ne permite un anumit joc, în aproximarea acestor contururi și deci, a încadrării unora dintre ele în curbe definibile matematic - deoarece nu toate structurile determinate sunt totodată și relevante, iar multe dintre ele nu sunt complet determinate. Pentru sistemul de percepție optică uman, recunoașterea formelor curbelor simple este, după cum am amintit în cazul iluziilor optice de cele mai multe ori o activitate neproblematică, ceea ce însă la ora actulă, pentru sistemele de calcul este în majoritatea cazurilor (încă) imposibilă.

Cu toate acestea, au fost și pe această cale făcuți pași deosebit de importanți, și care merită a fi pe scurt prezentați, cu atât mai mult cu cât ei sunt utilizați în cadrul lucrării de față în vederea generării de G-Code pentru prelucrarea ulterioară prin comanda numerică.

Structuri pregnante

În principal, putem deosebi două metode de recunoaștere a formelor simple utilizabile. Într-o primă variantă, s-ar putea, plecând de la un punct al conturului și urmărindu-i evoluția, să comparăm acest traseu cu conturul unor curbe cunoscute. Această metodă este însă anevoioasă și de cele mai multe ori duce la rezultate neclasificabile sau redundante.

O altă metodă, este cea de analizare globală a imaginii și extragerea structurilor pregnante aparținând unor forme cunoscute. Dintre acestea, deși tehnica permite doar o vagă introducere în ceea ce eventual va fi posibil peste câțiva ani, una dintre cele mai importante este transformarea Hough, algoritmul căreia a fost preluat, implementat, adaptat și îmbunătățit în cadrul lucrării de față în vederea utilizării sale în cadrul prototipării rapide.

Transformarea Hough pentru linii

Metoda a fost patentată de către Paul Hough în 1962 [36], fapt pentru care este cunoscută sub denumirea de "Hugh transformation" (HT) și este o metodă generală de localizare a curbelor parametrice în distribuții de puncte.

Metoda se bazează pe faptul că foarte multe curbe geometrice parametrice cu puțini parametri (linii, cercuri, elipse) reprezintă forme ale obiectelor generate de om.

Astfel, de exemplu în cazul unei linii, ecuația generală parametrică fiind

$$y = mx + n$$

și considerând faptul că o linie trebuie să treacă prin două puncte de coordonate $p_1(x_1, y_1)$ și $p_2(x_2, y_2)$, ecuațiile sistemului următor vor trebui să fie respectate.

$$y_1 = m x_1 + n$$
$$y_2 = m x_2 + n$$

deci, problema care se pune, este cea de găsirea variabilelor m și n $(m, n \in \mathbb{R})$ pentru care un număr maxim de puncte ale contururilor determinate rezolvă sistemul de mai sus. Deși determinarea tuturor liniilor posibile existente în imagine și examinarea apartenenței punctelor conturului la aceasta este posibilă, metoda ar fi ineficientă datorită numărului ridicat de linii posibile, fapt pentru care metoda va fi optimizată punând problema în mod invers, deci a determinării numărului de linii care trec printr-un punct

al conturului.

Spațiul parametrilor

Ținând cont de faptul că fiecare linie care trece prin punctul p_0 trebuie să respecte ecuația dreptei, vom avea:

$$L_j: y_0 = m_j x_0 + n_j$$

Alegând o valoare pentru m_j putem obține foarte simplu valoare lui n_j ca fiind

$$n_j = -x_0 m_j + y_0$$

deci tot o funcție liniară în care, de data aceasta variabilele sunt m_j și n_j, iar parametrii sunt coordonatele punctului de pe contur. Din acest motiv, acest spațiu (în cazul bidimensiunal fiind redus la un plan) poartă denumirea de "spațiul parametrilor".

Schematic, cele două spații vor arăta ca în figura următoare:

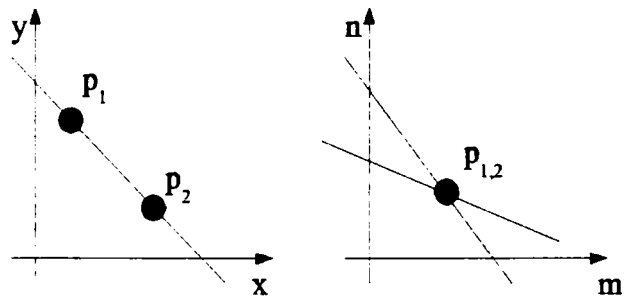


Fig. 170: Spații Hough imagine/parametrii

sau altfel spus, dacă N linii se intersectează într-un punct $P(m_n, n_n)$ în spațiul parametrilor, atunci în spațiul imaginii pe dreapta de ecuație $y = m_n x + n_n$ se vor afla N puncte.

Matricea de acumulare

Astfel, problema inițială a fost transformată într-una echivalentă, prin care dorim determinarea punctelor din spațiul coordonatelor în care se intersectează cât mai multe linii. Ideea lui Hough este dat de faptul că, utilizând o discretizare a spațiului parametrilor sub forma unei matrici, pentru fiecare punct de pe conturul din imagine va fi trasată în mod aditiv linia corespunzătoare în matricea de acumulare. Astfel, pentru fiecare element al matricii, în momentul în care acesta aparține uneia din liniile trasate, valoarea sa va fi incrementată. Schematic, principiul este reprezenta în figura de mai jos.

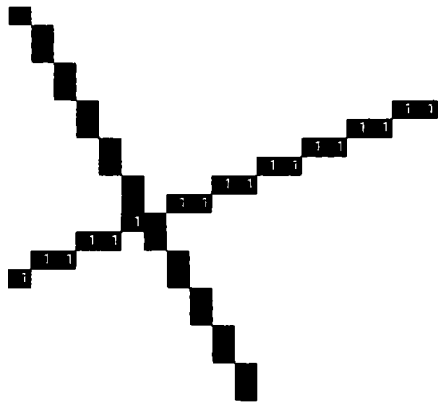


Fig. 171: Matricea de acumulare

Parametrizarea optimizată

Din nefericire, pentru verticale, în realitate vom obține valoarea $m=\infty$, fapt pentru care metoda nu este practicabilă. Din acest motiv, se recurge la așanumita "Forma normală Hesse" (HNF – Hesse Normal Form) de reprezentare a parametrilor

$$x\cdot\cos(\Theta)+y\cdot\sin(\Theta)=r$$

în care nu vom avea astfel de singularități.

Astfel, pentru un punct $P_i(x_i, y_i)$ vom obține relația

$$r_{x_i,y_i}(\Theta)=x_i\cdot\cos(\Theta)+y_i\cdot\sin(\Theta), \forall\,\Theta\in[0,\ \pi)$$

Astfel, considerând centrul imaginii ca originea sistemului de coordonate x/y, valoarea absolută maximă pe care o poate avea r este jumătatea diagonalei imaginii, sau

$$-r_{max}\le r_{x,y}(\Theta)\le r_{max}, r_{max}=\frac{1}{2}\sqrt{M^2+N^2}$$

în care am notat, ca mai înainte, dimensiunile imaginii cu M și respectiv N.

Cele două spații, vor avea o reprezentare asemănătoare imaginii care urmează.

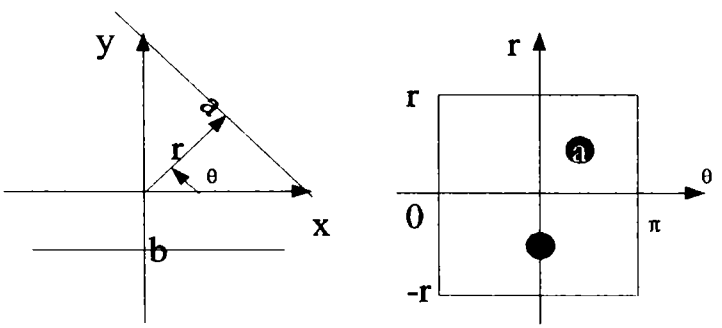


Fig. 172: Spațiul parametrilor HNF

Transformarea Hough pentru cercuri şi elipse

Dat fiind faptul că pentru ca un punct să fie aflat pe un cerc, el trebuie să rezolve ecuaţia

$$(x-x_c)^2+(y-y_c)^2=\rho^2$$

şi deci, în acest caz, spaţiul parametrilor va fi tridimensional.

Spre deosebire de cazul liniilor, pentru elipse şi cercuri nu există o dependenţă funcţională simplificată a coordonatelor din spaţiul parametrilor. O metodă de tip "brute force" este bineînţeles posibilă, însă utilizând observaţia că punctele prin aflate pe un cerc de rază dată pot fi construite cercuri de aceeaşi rază, iar acestea toate vor trece prin centrul cercului dat, nu vom mai fi nevoiţi să analizăm toate punctele din spaţiul parametrilor, ci doar să incrementăm valorile acumulatorului în fiecare din planele date de ρ.

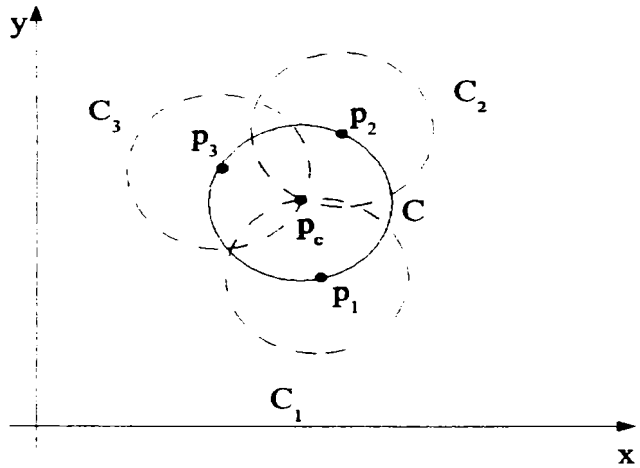


Fig. 173: Transformarea Hough pentru cercuri

Variind astfel domeniul razelor de interes de-a lungul celei de-a treia axe, vom obţine poziţia punctelor dorite.

În cazul elipselor, acestea fiind determinate de centrul elipsei, distanţele focale şi înclinare, vom avea un spaţiu 5-dimensional pentru determinarea obiectelor de acest tip, ceea ce duce la necesitatea de puteri de calcul ridicate. Pentru doar 128 de paşi (2^7) în fiecare dimensiune a spaţiului parametrilor vom avea nevoie de 2^{35} celule ale acumulatorului, ceea ce pentru valori întregi stocate în 4 bytes (integer4 = 2^2) vom avea nevoie de un spaţiu de memorie de $2^3 \cdot 2^2 = 2^{37}$ bytes, deci *128GB.*

Atât pentru acest caz, cât şi pentru alte forme 2D, se utilizează metoda Hough generalizată, prezentată pe larg în [37] şi [38], o versiune modificată a acesteia (adaptată pentru specificul analizei imaginilor în vederea prototipării rapide) fiind implementată în cadrul acestei lucrări.

Probleme şi rezolvări

În practică însă, chiar dacă principiile prezentate anterior sunt valabile şi funcţionabile, de cele mai nulte ori ele nu sunt suficiente sau satisfăcătoare pentru atingerea ţelului dorit. Pentru exemplificare, vom prezenta în continuare câteva exemple.

Bias

Datorită faptului că valorile acumulatorului conţin numărul de puncte prin care trece curba, acest număr este dependent de lungimea vizibilă, deci conţinută în imagine a curbei. Astfel, o linie de exemplu aflată în apropierea unui colţ al imaginii, şi deci de lungime redusă, va conţine mult mai puţine puncte decât una din apropierea diagonalei. Ţinând cont de faptul că noi vom căuta valorile maxime ale matricii, este posibil ca linia mai scurtă să fie ignorată.

Eroarea fiind sistematică, poate fi compensată prin normalizarea intrărilor din acumulator referitor la numărul maxim posibil de valori posibile. Notând cu $A[\theta, r]$ valoarea acumulatorului şi cu $A_{max}[\theta, r]$ numărul maxim posibil vom obţine valoarea normalizată

$$A'[\theta, r] = \frac{A[\theta, r]}{A_{max}[\theta, r]}$$

Puncte de capăt

După cum am observat, transformarea Hough ne permite în implementarea amintită mai sus doar determinarea liniei, dar nu şi a poziţiei punctelor de început şi de sfârşit ale acesteia. O determinare ulterioară a acestora este, de cele mai multe ori datorită erorilor de rotunjire, imprecisă şi deci nesatisfăcătoare.

Pentru compensarea acestei erori şi concomitent a obţinerii punctelor amintite, vom extinde conţinutul fiecărei celule a acumulatorului cu coordonatele de început şi sfârşit.

Astfel, valorile matricii vor avea forma

$$A[\theta, r] = (n, X_{start}, Y_{start}, X_{end}, Y_{end})$$

Conţinutul valorilor suplimentare va fi actualizat pentru fiecare punct întâlnit, astfel încât la sfârşitul scanării imaginii acestea să conţină valoarea celor mai îndepărtate puncte localizate. Deşi expandarea acumulatorului în vederea stocării informaţiei capetelor liniilor n-a fost implementată, adoptarea acestei metode (eventual combinată cu metoda Harris prezentată anterior) are o valoare inestimabilă în prototiparea rapidă, permiţând deplasări prin iterpolare liniară (G1) între aceste puncte, acet tip de interpolare fiind avantajos cel puţin din următoarele motive:

- interpolarea liniară pe maşină este calitativ superioară celei calculate prin scalarea punctelor imaginii
- regimul de aşchiere este constant pe toată durata prelucrării cu excepţia apropierii de punctele de capăt, pe când în cazul succesiunii de interpolări liniare acesta fiind sacadat
- secvenţele de G-Code generate sunt comprimabile, flexibile şi optimizabile

În cadrul lucrării de faţă s-a recurs însă la o metoda mai simplă şi care necesită atât putere de calcul mai redusă cât şi o durată de timp mult mai mică, prin utilizarea unui algoritm de postprocesare a vecinătăţilor punctelor. Metoda teoretică prezentată acum însă este evident calitativ superioară, prin faptul că unghiurile astfel determinate nu sunt constrânse la valorile unghiurilor date de vecinătăţi.

Considerarea calităţii şi orientării marginii

După cum am aflat anterior, în urma detectării muchiilor obţinem concomitent şi o valoare corespunzătoare dimensiunii muchiei (valoare pe care în continuare o vom nota cu $E(u, v)$), valoare care este uşor de luat în considerarea detectării curbelor simple, prin simpla adăugare a valorii ei la valoarea acumulatorului, dat fiind faptul că pentru o muchie de calitate ridicată are o dimensiune redusă. Astfel, valoarea acumulatorului va fi dată de

$$A[\theta, r] = A[\theta, r] + E[u, v]$$

Valoarea orientării poate fi şi ea utilă, aşa cum este descris în [39].

Transformări ierarhice

Precizia rezultatelor creşte evident cu mărimea acumulatorului. Rezoluţia unghiulară a transformării este dată de formula:

$$R = \frac{\pi}{n}$$

Astfel, pentru 256 de valori vom avea o rezoluţie unghiulară de $\frac{\pi}{256} \approx 0{,}7°$.

Pe de altă parte însă, mărirea acumulatorului duce la o creştere a puterii de calcul necesară, fapt pentru care în practică se poate recurge la ierarhizarea transformării, deci aplicarea unei măriri (zoom) a spaţiului parametrilor în urma unei determinări grobe a poziţiei curbelor. Prin această operaţie, poziţia curbei va fi mult mai exact determinată, rezoluţia fiind repartizată pe o zonă mai mică a domeniului.

5.2.3.8 Regiuni în imaginile binare

Ţinând cont de faptul că în cazul imaginilor binare separarea fundalului de obiectele conţinute de imagine este cea mai simplă, se pune problema izolării acestor obiecte. Luând în considerare faptul că suntem nevoiţi să determinăm forma obiectelor precum şi numărul acestora, lucru care prin simpla analiză a punctelor componente ale imaginii nu este posibil, fiind necesară gruparea punctelor aparţinătoare fiecărui obiect. Astfel, în primul rând este foarte important să grupăm punctele adiacente aparţinătoare unui obiect.

Metode de determinare a regiunilor

Înainte de a reuşi să determinăm obiectele din cadrul imaginii, este esenţială determinarea regiunilor imaginii în care pixelii care aparţin acestuia, process care poartă denumirea de "region labeling" sau "region coloring". Ambele denumiri se referă la acelaşi proces, intuitiv acest lucru fiind explicabil prin asocierea unui număr corespunzător unei nuanţe de gri sau unei culori fiecărui punct aparţinând regiunii curente.

Variantele cele mai cunoscute le vom prezenta în continuare.

Flood Filling

Principiul de funcţionare al acestui proces, este asemănător celui prezentat în cazul algoritmului "Grassfire". Singura deosebire rezidă în faptul că, pentru alegerea punctului de pornire se recurge iniţial la căutarea unui pixel care n-a fost analizat până în momentul respectiv. Diagrama de activitate corespunzătoare este prezentată în Fig. 174.

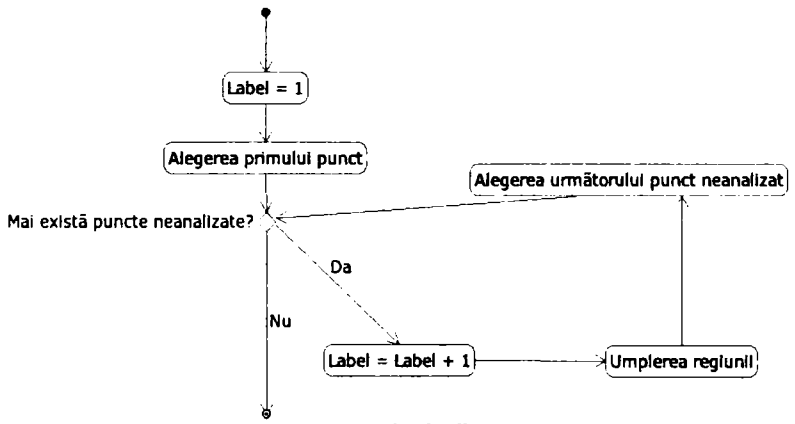


Fig. 174 Flood Filling

În principiu există trei metode implementabile ale algoritmului:

- Recursiv
- Iterativ "depth first"
- Iterativ "breath first"

Prima variantă este simplu de implementat, în schimb este utilizabilă pentru imagini de dimensiune redusă, datorită faptului că în vederea recursivității, înaintea apelării procedurii recursive, datele utilizate în procedură trebuie salvate pe aşanumitul "call stack", acesta crescând cu fiecare nivel de recursivitate (în general pînă la imagini de 200×200 pixeli).

Înlocuind stack-ul intern cu un stack programabil, orice procedură recursivă poate fi implementată într-un mod iterativ. Diferența este în faptul că acest tip de stack este limitat de memoria calculatorului și poate fi dinamic alocat (în Heap).

Marcarea secvențială

Marcarea secvențială este o metodă clasică, nerecursivă, constituită din două etape:

- marcarea preliminară
- eliminarea conflictelor de marcaj

Marcarea preliminară

Imaginea este inițial scanată de la stânga sus spre dreapta jos în funcție de vecinătatea aleasă utilizându-se matricile următoare:

	N_2	
N_1	X	

Fig. 175 Marcare - vecinătate tip 4

N_2	N_3	N_4
N_1	X	

Fig. 176 Marcare - vecinătate tip 8

În Fig. 177 este prezentată stilizat, pentru o explicație mai clară a modului de funcționare al algoritmului, o imagine digitală binară.

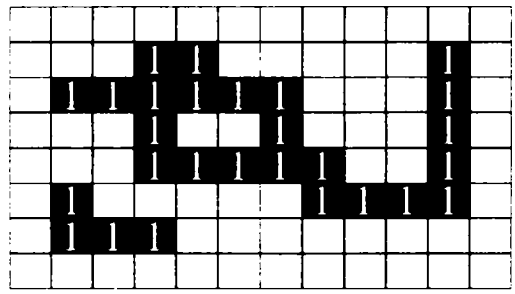


Fig. 177 Marcare - imagine stilizată

Scanarea punct cu punct începe în colțul din stânga sus a imaginii și este continuată fără evenimente până în momentul în care este întâlnit primul pixel de valoare 1 care n-are nici un vecin găsit până acum (Fig. 178).

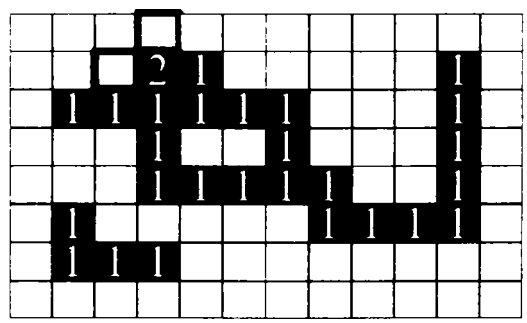


Fig. 178 Marcare - Primul punct

Valoarea acestui pixel va fi primi din acest moment eticheta 2, iar căutarea continuă. Următorul pixel de valoare 1 va avea un singur vecin, a cărui valoare este 2, valoare care va fi preluată după cum se vede din figura următoare..

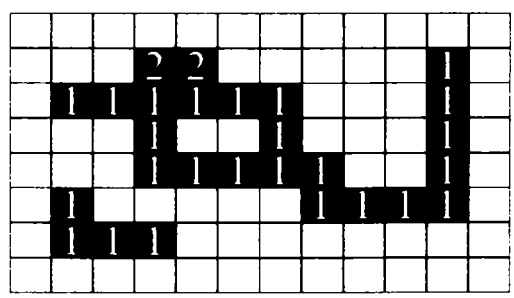


Fig. 179 Marcare - prima preluare

Punctul următor întâlnit este cel notat în Fig. 180 cu 3, care neavând nici un vecin determinat, va primi următoarea valoare a etichetei (valoarea anterioară fiind 2). În mod identic vom obține valorile notate cu 4. În aceeași figură, observăm că în momentul analizării pixelului cu valoarea 1, valorile pixelilor vecini sunt 2 și 4, motiv pentru care vom denumi acest caz coliziune, deoarece ambele regiuni găsite până acum aparțin de fapt aceleiași regiuni și deci ar trebui să dețină aceeași etichetă. Vom păstra din acest motiv în memorie faptul că valorile de 4 vor trebui înlocuite cu valoarea 2.

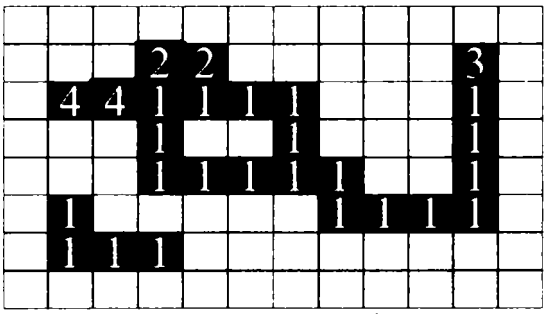


Fig. 180 Marcare - Prima coliziune

În același timp, valoarea actuală a marcajului va fi, evident, de asemenea 2, după cum se observă din figura următoare, coliziunea fiind preliminar rezolvată, fără eliminarea duplicatelor etichetelor, acestea fiind însă memorate.

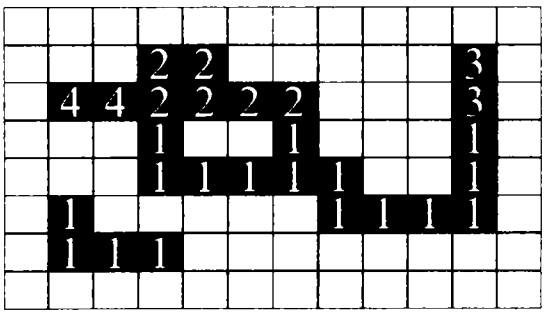


Fig. 181 Marcare - Rezolvarea preliminară a coliziunii

Odată cu baleierea completă a imaginii, exemplul inițial va avea următoarea reprezentare, zonele de conflict fiind de asemenea marcate.

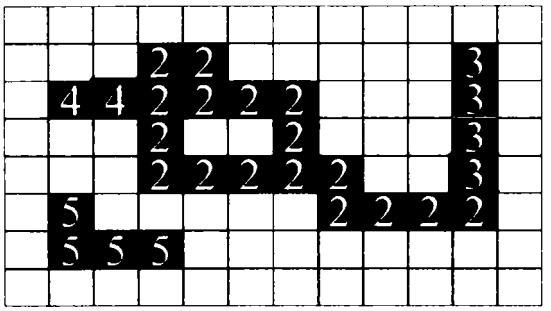


Fig. 182 Marcare - Sfârșitul fazei preliminare

Eliminarea conflictelor

Schematic, putem reprezenta lista de coliziuni ca în figura următoare. Astfel, în cazul prezentat anterior, eticheta cu numărul 2 va înlocui cele de valoare 3 și 4.

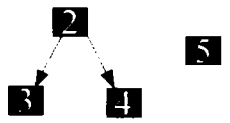


Fig. 183 Marcare - Lista coliziunilor

În urma efectuării înlocuirilor, deci a eliminării conflictelor, vom obține imaginea separată prezentată în continuare.

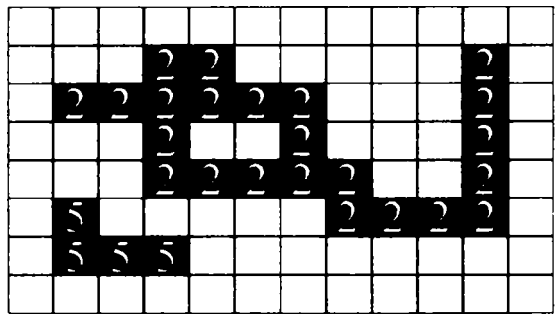


Fig. 184 Marcare - imaginea separată

5.2.3.9 Segmentarea

Țelul de bază al segmentării este separarea obiectelor de interes din cadrul imaginii analizate în vederea prelucrării lor ulterioare. Prin această caracteristică, segmentarea este unul dintre cei mai importanți pași al prelucrării automate a imaginilor, deoarece numai în acest mod obiectele detectate pot fi măsurate și identificate.

De exemplu, în cazul textelor scanate, este importantă separarea literelor în vederea interpretării lor, iar în cazul analizării imaginilor aeriene sau din satelit, este de importanță crucială separarea caselor, a pădurilor, străzilor, apelor curgătoare și a lacurilor, a cîmpiilor și stâncilor.

Prin segmentare sunt separate obiecte de fundal sau obiecte singulare dintr-un group și se bazează pe metodele prezentate anterior.

Metodele aplicabile sunt și în acest caz, ca în cazurile prezentate anterior numeroase, cele mai cunoscute și importante fiind prezentate în continuare.

Metode orientate pe puncte

Cele mai simple metode sunt bineînțeles și în acest caz cele bazate pe puncte și se bazează pe apartenența pixelului curent la obiectul actual. Un rol deosebit de important în vederea segmentării îl joacă histogramele.

Metode în trepte

Trepta simplă

În cazul metodelor în trepte, segmentarea se face după funcția de transformare:

$$P_T(u,v)=\begin{cases}1, pentru\ I(u,v)\geq T\\ 0, pentru\ I(u,v)<T\end{cases}, \qquad 0\leq T\leq T_{max}$$

unde cu T_{max} am notat valoarea maximă pe care o poate avea treapta (1 pentru imagini binare, 255

pentru imagini gri).

Astfel, pe baza histogramei valorilor de gri, este posibilă separarea în două nivele a oricărei imagini digitale. În imaginile următoare este reprezentată separarea utilizând o treaptă de 64 și una de 128.

Calculul optimal al treptei

Una din problemele principale ale segmentării este faptul că în cadrul histogramelor nu poate fi făcută di-

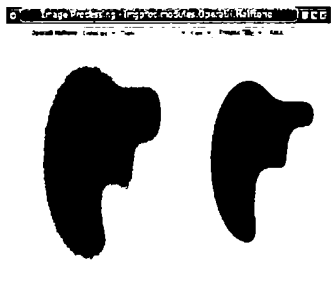


Fig. 185 Segmentare - trepte simple 64

Fig. 186 Segmentare - trepte simple 192

ferența între valorile de gri ale obiectului dorit și cele ale fundalului. Din acest motiv ne vom folosi de funcția de densitate a probabilității atât a obiectului (P_O) cât și a fundalului (P_B) . O foarte bună aproximație pentru această funcție este dată de funcția de densitate a probabilității dată de Gauss $p_i(z)$:

$$p_i(z)=\frac{1}{\sqrt{2\pi}\sigma_i}e^{-\frac{(z-\mu_i)^2}{2\sigma_i^2}} \qquad ,i=O,B$$

unde μ_i sunt mediile iar σ_i abaterile standard pentru probabilitățile apriori P_O și P_B și bineînțeles

$$P_O+P_B=1$$

Astfel, repartizarea intensităților pe toată imaginea este dată de

$$p(z)=P_O\,p_O(z)+P_B\,p_B(z)$$

După acest principiu, putem ordona punctele imagini ca aparținând fundalului și cele aparținând obiectului doar cu un anumit grad de eroare, dat fiind faptul că repartiția probabilităților se suprapune parțial. Acest grad de eroare, este dat de expresia:

$$E_B(T)=\int_{-\infty}^{T} p_O(z)\,dz$$

și de asemenea, gradul de eroare al clasificării punctului ca aparținând obiectului:

$$E_O(T)=\int_{T}^{\infty} p_B(z)\,dz$$

împreună cu eroarea totală:

$$E(T)=P_O E_B(T)+P_B E_O(T)$$

Pentru determinarea valorii optimale a valorii lui T, vom minimiza funcția erorii prin derivare și determinarea punctelor în care valoarea derivatei este zero.

$$(\sigma_B^2-\sigma_O^2)T^2+2(\mu_B\sigma_O^2-\mu_O\sigma_B^2)T+\sigma_B^2\mu_O^2-\sigma_O^2\mu_B^2+2\,\sigma_B^2\sigma_O^2\ln\left(\frac{\sigma_O P_B}{\sigma_B P_O}\right)=0$$

Considerând pentru simplificare că $\sigma_O=\sigma_B=\sigma$, vom obține

$$T=\frac{\mu_O+\mu_B}{2}-\frac{\sigma^2}{\mu_O-\mu_B}\ln\frac{P_O}{P_B}$$

Trepte locale

În cazul în care eroarea este ridicată, imaginea poate fi împărțită în subimagini, iar acestea la rândul lor vor fi supuse analizei amintite anterior, evitându-se în acest caz aproximarea eronată a erorilor datorită domeniului mare al imaginii și fluctuațiilor mari de valori.

5.2.3.10 EDISON (Edge Detection and Image SegmentatiON)

Datorită faptului că detectarea contururilor și segmentarea au loc simultan în cadrul acestui algoritm [40], programul a fost adaptat la cerințele necesare atingerii scopului nostru – aplicarea lui în cadrul prototipării rapide.

Descrierea algoritmului

Algoritmul este descris în [41], dar punctele principale le vom aminti în continuare.

După cum am vazut, algoritmii practicabili utilizează fie metode în trepte, fie bazate pe domenii de medii (k-mean clustering) în cazul determinării optimale a treptelor. Unul dintre dezavantajele cele mai mari ale algoritmilor de ceast tip este faptul ca valoarea de segmentare (k) trebuie sa fie cunoscută a priori. Prin metoda prezentată, bazată pe deplasarea mediei (mean-shift), acest dezavantaj este eliminat, în defavoarea puterii de calcul necesare. Tocmai datorită faptului că ideile prezentate în această lucrare se bazează pe sisteme distribuite, acest impediment devine unul dintre avantajele sistemului. Astfel, imaginea poate fi cu ușutință despărțită în mai multe imagini separate, distribuite în cluster și rezultatele reordonate în imaginea rezultată finală.

Deși algoritmul este complicat, utilizarea pachetului este fară mari modificări aplicabilă scopului lucrării, deoarece detectarea muchiilor este foarte flexibilă, rezultatul poate fi cu ușurință implicat în determinarea liniilor și cercurilor, caracteristici geometrice importante pentru prelucrarea cu comenzi numerice. Evident că utilizarea altor algoritmi de generarea a comenzilor numerice bazate pe simpla urmărire punct cu punct a contururilor depistate (de asemenea implementat printr-un exemplu în cadrul acestei lucrări), dar eficiența utilizării burghielor și frezelor precum și generarea comenzilor de interpolare circulară necesită determinarea caracteristicilor amintite anterior.

Exemplu de aplicare

În vederea detectării contururilor, vom utiliza imaginea digitală următoare:

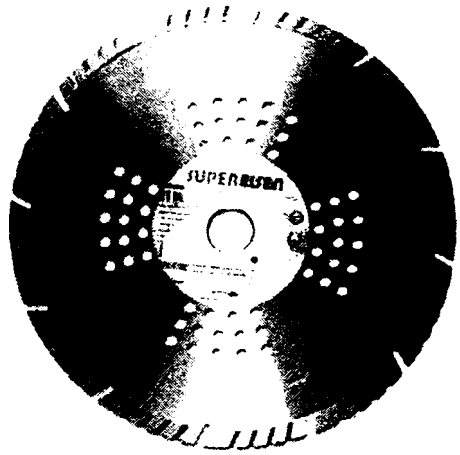


Fig. 187: Imaginea de bază în vederea segmentării şi detecţiei muchiilor

În urma aplicării algoritmului de segmentare şi a celui de detecţie a muchiilor, vom obţine imaginile prezentate în continuare.

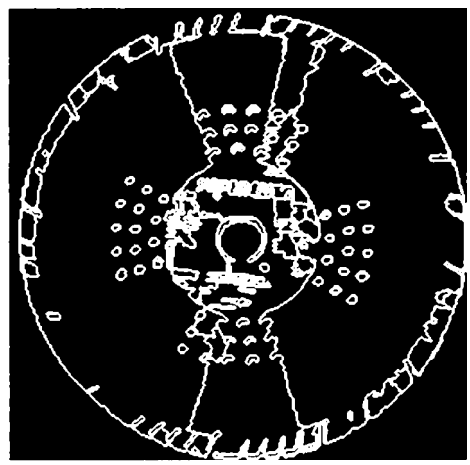


Fig. 188: EDISON - detectarea muchiilor

5.2.3.11 SUSAN (Smallest Univalue Segment Assimilating Nucleus)

Un alt algoritm foarte eficient de segmentare, prezentat este în [42]. Algoritmul are marele avantaj, pe lângă viteza sa de lucru, al unei scalări liniare a numărului de cante determinate în raport cu timpul. Deşi greu aplicabil în aplicaţii distribuite datorită dificultăţii determinării zonelor aderente în momentul recompunerii imaginii rezultate, algoritmul este demn de a fi descris în continuare.

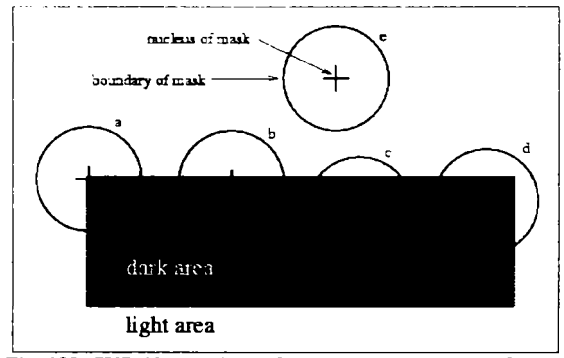


Fig. 189: SUSAN - măşti circulare într-o imagine exemplară

Descrierea algoritmului

Principiul algoritmului este de dete.....e ..a.- punsului funcţiei imaginii digitale inversate astfel încât cantele să fie evidenţiate, iar colţurile şi mai puternic accentuate. În acest scop vor fi utilizate ca mască aşa numite nuclee, reprezentate prin cercuri. [43]. Luminozitatea fiecărui pixel din mască este comparată cu cea a nucleului. Figura următoare reprezintă câteva cazuri caracteristice.

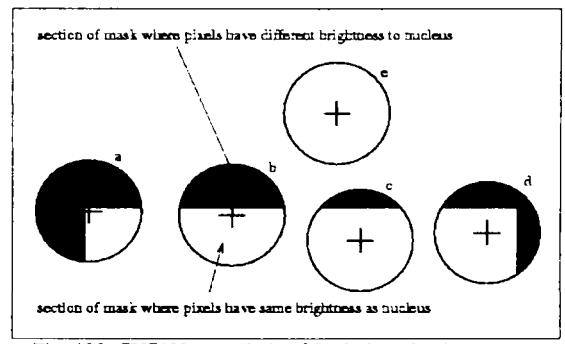


Fig. 190: SUSAN - măşti circulare în imagine inversată

Aceeaşi imagine, inversată este reprezentată în . În această imagine, zonele USAN (Univalue Segment Assimilation Nuceus) ale măştilor din figura anterioara sunt reprezentate în alb.

Suprafaţa unui USAN, după cum reise din imaginea anterioară, este maximă în cazul în care masca es e complet ntr-o reg une fara muc , sca e în momentul conţinerii unei muchii şi scade şi mai mult în momentul conţinerii unui colţ. Exemplul din figura următoare clarifică principiul.

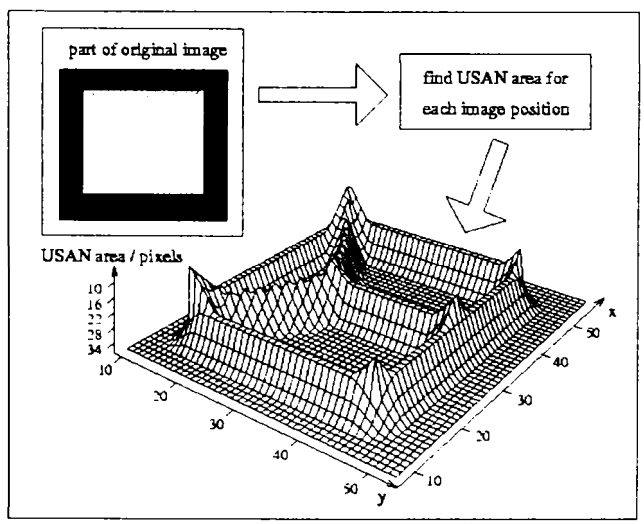


Fig. 191: SUSAN - exemplu simplu

Exemplu de aplicare

Aplicând algoritmul asupra imaginii de analizat (Fig. 187), vom obține rezultatul următor:

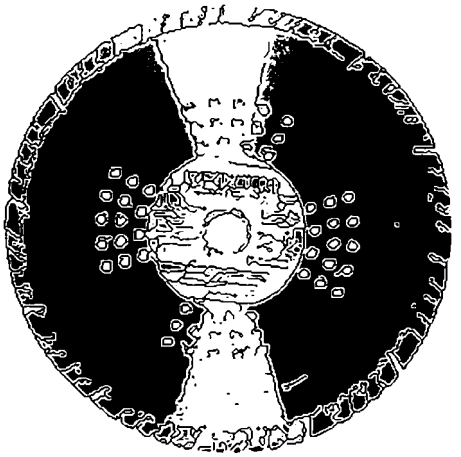


Fig. 192: SUSAN - detectarea muchiilor

5.2.3.12 Efficient Graph-Based Image Segmentation

Algoritmul cel mai potrivit în vederea atingerii scopului lucrării este considerat cel prezentat în continuare. Avantajul principal este cel dat de faptul că sursele C++ ale utilitarului sunt publice iar modelul matematic a fost prezentat în [44]. Cu toate că formatul utilizat este doar PPM, prin existența a numeroase altor utilitare care permit conversia în batch a imaginilor în acest format, impedimentul poate fi ignorat.

Descrierea algoritmului

Ținând cont de faptul că descrierea pe larg atât a modelului matematic cât și a algoritmului este pe larg prezentată în carul surse mai sus amintite, vom recurge în continuare doar la o scurtă descriere a algoritmului, doar în vederea clarificării modului sau de funcționare și a avantajelor sale.

Așa cum îi spune și numele, metoda este bazată pe teoria grafurilor, utilizând grafuri nedirecționate atât pentru vertice cât și pentru cante, acestea din urmă conținând ca valoare pozitivă măsura diferenței dintre cele două vertice pe care de unește (pondere). În cazul imaginilor digitale, verticele sunt reprezentate de pixeli, iar ponderea este diferența uneia sau mai multor proprietăți dintre pixelii învecinați (culoare, intensitate, poziție, etc.).

Astfel, segmentarea va fi redusă la determinarea și separarea pixelilor în zone, astfel încât fiecare zonă să corespundă unui graf conectat, sau altfel spus, fiecare segment este indus printr-un subset de margini inclus în imaginea inițială, astfel încât ponderile marginilor dintre pixelii aceleiași zone să fie de valori reduse, pe când cele dintre zone diferite au valori corespunzătoare foarte mari.

Exemplu de aplicare

Aplicând algoritmul asupra imaginii de analizat (Fig. 187), vom obține rezultatul următor:

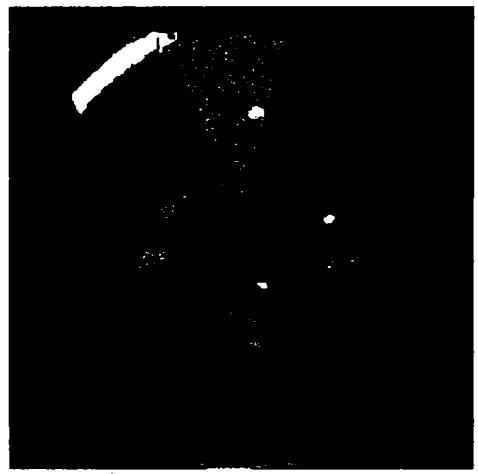


Fig. 193: Efficient Graph-Based Image Segmentation

5.2.3.13 Extracția proprietăților

În vederea luării de decizii calitative sau cantitative este necesară extragerea informațiilor semantice specifice imaginii analizate.

În viața reală, extragerea acestor informații, prelucrarea lor și luarea deciziilor referitoare la clasarea obiectelor observate pare a fi nespus de simplă. În momentul în care încercăm însă să modelăm acest proces, lucrurile devin deosebit de complicate, implicând probleme tehnice și matematice ridicate, pe care le vom aminti în continuare pe scurt.

Proprietăți geometrice

Dintre proprietățile cele mai importante din punctul de vedere al scanării în vederea prototipării rapide sunt bineînțeles cele geometrice – deci legate de formă, mărime și poziție.

Proprietăți geometrice globale

Dintre proprietătile principale ale obiectelor binare putem aminti

- suprafața
- perimetrul
- factorul de formă
- centrul geometric de greutate
- raportul razelor

Suprafața

Ținând cont de faptul că o imagine binară cu M coloane și N linii are forma:

$$I(u,v), u=0 \ldots, M-1,\ v=0, \ldots, N-1$$

Putem nota suprafața unui obiect al imaginii după cum urmează:

$$A=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} I(u,v), I(u,v)=\{0,1\}$$

unde am considerat că în cadrul imaginii doar obiectul de analizat este reprezentat.

Perimetrul

În vederea obținerii perimetrului, vom aplica transformarea binară

$$B=(I\oplus H_n)/I$$

Astfel, perimetrul obiectului poate fi calculat după formula:

$$P=\sum_{u=0}^{M}\sum_{v=0}^{N} B(u,v), B(u,v)=\{0,1\}$$

Pentru o mai precisă calculare, vecinătățile immediate pot fi ca având valoarea 1 iar vecinii 'îndepărtați' cu valoarea de $\sqrt{2}$.

Factorul de formă

Raportul dintre patratul perimetrului și suprafață poartă numele de factor de formă, notat așadar prin:

$$F=\frac{P^2}{4\pi A}$$

deci, pentru un obiect circular, acest raport este 1.

Centrul geometric de greutate

Centrul geometric de greutate al unui obiect al imaginii binare este dat de:

$$u_c=\frac{1}{A}\sum_{i=0}^{A} u_i$$

$$v_c=\frac{1}{A}\sum_{i=0}^{A} v_i$$

Raportul razelor

Bazat pe centrul de greutate al obiectului de analizat și notând cu R_{max} raza maximă a cercului circumscris cu centrul în centrul de greutate și cu R_{min} al celui înscris, se poate defini raportul acestor raze după cum urmează:

$$R=\frac{R_{max}}{R_{min}}$$

Reprezentarea Fourier a contururilor

Proprietăţile amintite anterior caracterizează obiectul printr-un singur număr, aşadar informaţiile despre obiect sunt pierdute. Utilizând o transformare discretă Fourier însă, se poate obţine o caracterizare cantitativă a obiectului subiect. În acest scop, vom utiliza punctele de pe perimetrul obiectului pentru a defini un nou obiect, care în general poate fi aproximat printr-o serie Fourier cu un număr redus de coeficienţi. Astfel, considerând obiectul perimetrului O_P fiind constituit din puncte de coordonate u şi v, putem defini poziţia fiecărui punct printr-o valoare complexă

$$p(n)=u(n)+iv(n),\ 1\le n\le P$$

Momente în imagini

O altă proprietate geometrică ce caracterizează un obiect este momentul, definit prin:

$$m_{pq}=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} I(u,v)u^p v^q,\ p,q=0,1,2\ldots$$

în care $(p+q)$ poartă numele de 'Gradul Momentului'.

Centrul de greutate al obiectului se va calcula deci prin formula:

$$\begin{cases} u_c=\dfrac{m_{10}}{m_{00}} \\ v_c=\dfrac{m_{01}}{m_{00}} \end{cases}$$

Pornind de la aceste coordonate, pot fi definite momentele centrale de ordin superior, notate cu

$$\mu_{pq}=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} I(u,v)(u-u_c)^p(v-v_c)^q$$

Normalizând acest moment central, se vor obţine alte şapte momente care au însă proprietatea de a fi invariante din punct de vedere al tranzaţiei, rotaţiei şi scalării, lucru foare important în vederea clasificării ulterioare a obiectelor analizate. Momentul central normalizat este definit prin:

$$\eta_p=\frac{\mu_{pq}}{\mu_{00}^{\frac{p+q}{2}+2}}$$

iar cele şapte momente amintite sunt:

$$\Phi_1 = \eta_{20} + \eta_{02}$$
$$\Phi_2 = (\eta_{20} - \eta_{02})^2 + 4\,\eta_{11}^2$$
$$\Phi_3 = (\eta_{30} - 3\,\eta_{12})^2 + (3\,\eta_{21} - \eta_{03})^2$$
$$\Phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$
$$\Phi_5 = (\eta_{30} - 3\,\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)$$
$$\Phi_6 = (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$
$$\Phi_7 = (3\eta_{21} - \eta_{30})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)$$

5.2.4 Caracterizarea

Prin caracterizare se urmăreşte în primul rând o rată ridicată de compresie a datelor. Din matricea imaginii, se extrage un vector caracteristic ce conţine informaţiile importante ale obiectelor din imagine. Aceste informaţii sunt evident caracteristice aplicaţiei curente, ele stând la baza clasificării.

5.2.5 Clasificarea

Prin clasificare se urmăreşte repartizarea unui obiect detectat unei clase de obiecte cunoscute. Din acest motiv, pentru clasificare este necesară existenţa de clase de obiecte, reprezentate prin caracteristicile lor specifice (aceste fiind de cele mai multe ori determinate prin analiza unor probelor caracteristice). Ca rezultat, este obţinută o listă de elemente caracteristice cu ponderi şi factori de evaluare.

O metodă adecvată de generare a claselor este prin utilizarea de algoritmi de învăţare controlată, în cazul în care o determinare vizuală fără echivoc a obiectelor este posibilă.

În urma segmentării şi extragerii proprietăţilor caracteristice, pasul decisiv este cel al clasificării obiectelor, deci a încadrării acestora în clase de obiecte. Operaţia poartă deseori şi denumirea de Etichetare [45].

Dat fiind faptul că în urma etichetării obţinem numeroase proprietăţi caracteristice obiectului reprezentat, sarcina principală este cea de a determina cele relevante în vederea clasificării. Astfel, în cazul în care am reuşit să determinăm c caracteristici relevante ale unui tip de obiect, pe care le vom păstra într-un vector corespunzător, toate obiectele aparţinând acestei clase se vor poziţiona într-o zonă a spaţiului atributelor. Singura activitate care ne desparte de clasificarea obiectelor devine astfel separarea acestor grupe.

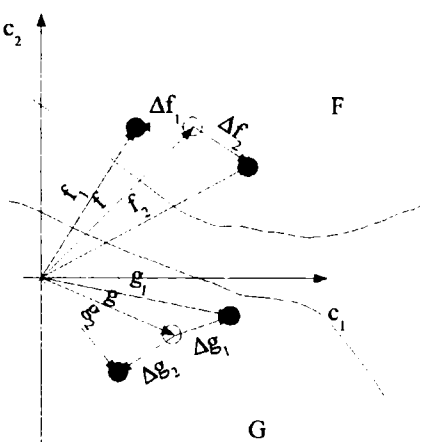


Fig. 194: Clasificare

Considerând astfel o simplă reprezentare vectorială 2D a unei clase, schematic clasificarea poate fi ilustrată în modul următor:

Astfel, considerând două clase F şi G alse căror obiecte reprezentative sunt reprezentate de vectorii caracteristici f şi g, precum şi obiectele de clasificat

f_1, f_2, g_1 și g_2, clasificarea acestor obiecte, ca aparținând uneia sau alteia din clase, este realizată fie prin determinarea liniilor de separare a domeniilor respective, fie prin încadrarea obiectelor într-un domeniu Δ care să cuprindă toate valorile Δf și respective Δg posibile. Din acest motiv, este evident că vom avea parte de numeroase probleme de clasificare. Aceste probleme sunt foarte numeroase, cele mai importante fiind:

- domeniile de clasificare nu sunt clar definibile
- clasele de încadrare nu sunt complet separabile
- obiectele reprezentative nu au fost adecvat alese
- caracteristicile alese nu sunt întradevăr reprezentative

Din acest motiv, metodele de clasificare sunt foarte variate, fiecare metodă încercând pe de o parte să elimine dezavantajele celeilalte, iar pe de altă parte specializându-se pe anumite tipuri de obiecte sau domenii de utilizare.

5.2.5.1 Metode de clasificare

Dintre cele mai utilizate metode de clasificare vom aminti:

- sistemele expert
- rețelele neuronale
- metode numerice de clasificare

Sisteme expert

Sistemele expert pot fi considerate ca o abstractizare direcționată a sistemelor de baze de date, prin care se urmărește selectarea unei informații speciale bazat pe o vastă bază de informații acumulate anterior și stocate într-o așanumită bază de cunoștințe.

Din acest motiv, aceste metode pot fi utilizate doar în domenii în care există cel puțin modele și/sau teorii (reguli) sau în cazul în care baza de cunoștințe aferentă este foarte vastă, bazată pe experiența în general a unui număr mare de experți. Prin aplicarea regulilor existente este astfel posibilă extragerea informației dorite, iar prin generarea de noi reguli, este posibilă obținerea chiar și a unui proces de învățare.

Rețele neuronale

Prin utilizarea modelului de funcționare a sistemului nervos, s-a ajuns la implementarea rețelelor neuronale, bazate pe capacitatea acestora de a lua decizii în urma analizării răspunsului unui număr foarte mare de elemente contactoare (neuroni). Astfel de rețele au marele avantaj (observabil din natură), că sunt foarte receptibile și adaptabile la apariția de condiții noi, deci a procesului de învățare.

Un alt avantaj deosebit de important este cel al vitezei de lucru și a stabilității (toleranța erorilor).

Chiar dacă cercetările din acest domeniu sunt încă la început, în ultimii ani s-au obținut în acest domeniu rezultate surprinzătoare.

Metode numerice de clasificare

Aceste metode clasice, bazate pe elemente geometrice și statistice, este cel mai des utilizate la ora actuală în vederea clasificării, dar în general sunt aplicabile în cazuri speciale și prin intervenția succesivă a unui operator uman.

5.2.5.2 Alegerea proprietăților

După cum am văzut, numărul de proprietăți sau atribute caracteristice ale unui obiect poate fi foarte mare, ceea ce va aduce cu sine creșterea dimensiunii vectorului acestor proprietăți și implicit prin numărul de operații necesare efectuate asupra sa, a timpului clasificării. Astfel, în cazul în care au fost detectate 300 de caracteristici, însă doar 30 vor fi utilizate în vederea clasificării, numărul de posibilități de alegere a acestor caracteristici este de

$$C_{300}^{30} \approx 1{,}7 \cdot 10^{41}.$$

Este deci clar, privind exemplul anterior, că alegerea unui set de atribute caracteristice reprezentative este departe de a fi trivială. Din acest motiv, în primul rând se va recurge la stabilirea unui nivel de calitate a caracteristicilor iar apoi a unei metode de selecție a acestora.

Criterii de calitate

Considerând că avem m elemente caracteristice independente din punct de vedere statistic, din care vom fi nevoiți să alegem cele mai bune n, situația este relativ simplă, deoarece vom putea alege un criteriu de calitate prin care diferența de criterii poate fi eliminată.

Valoarea medie și dispersia

Plecând de la considerentul că având clasele C_i și notând cu c_v o anumită caracteristică, precum și faptul că valorile mediei μ_{iv} și dispersiei σ_{iv} specifice clasei sunt cunoscute, măsura calității acestei caracteristici în două dintre clase este dată de

$$Q_{ijv} = \frac{(\mu_{iv} - \mu_{jv})^2}{\sigma_{iv}^2 + \sigma_{jv}^2}$$

așadar, o caracteristică a unei clase la care valoarea medie este cât mai îndepărtată de cea corespunzătoare a celei de-a doua clase și/sau suma dispersiilor este maximă, este adecvată pentru separarea celor două clase. Schematic, acest principiu este prezentat în Fig. 195:

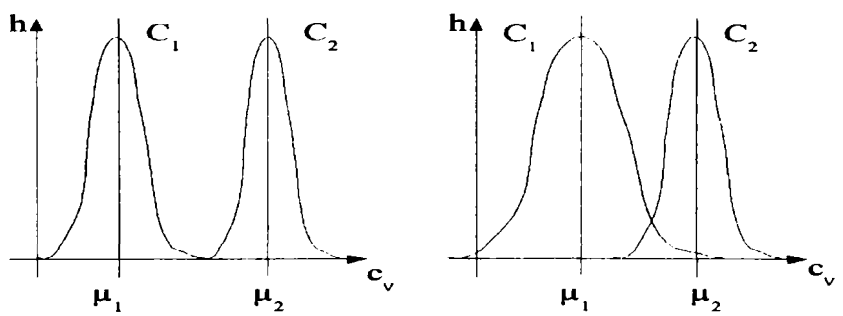


Fig. 195: Criterii de calitate

Distanța medie pătratică

În cazul în care cele două clase ce urmează a fi separate au caracteristici ale căror valori statistice sunt reprezentabile ca în cazul din figura următoare, deci pentru distribuții nenormale, este utilizabilă distanța medie pătratică.

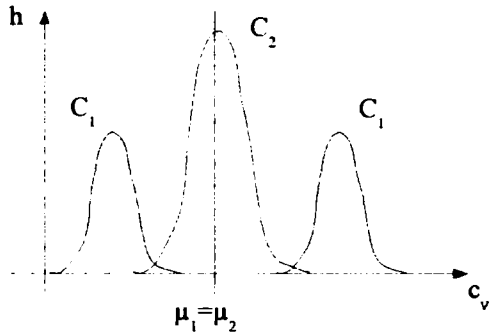


Fig. 196: Distanța medie pătratică

Distanța medie pătratică este dată de formula:

$$Q_{ijv} = \int_{c_v=-\infty}^{\infty} \left[h\{c_v|C_i\} - h\{c_v|C_j\} \right]^2 dc_v$$

unde $h\{c_v|C_k\}$ este probabilitatea de repartiție a densității caracteristicii c_v în clasa C_k.

Funcția de ambiguitate

Această funcție prezentată în [46] duce la obținerea unei mărimi prin care mărimea puterii de separare a caracteristicii este pusă în evidență, și este dată de formula:

$$A = -\sum_{j=1}^{a} \sum_{i=1}^{b} p(c_j)\, p\{C_i|c_j\} \log_b p\{C_i|c_j\}$$

în care am notat cu

- a – numărul de intervale
- b – numărul de clase
- $p(c_j)$ – probabilitatea apartenenței caracteristicii c_j în intervalul j
- $p\{C_i|c_j\}$ - probabilitatea apartenenței în clasa C_i în cazul utilizării caracteristicii c_j

Singura condiție impusă este ca numărul de probe efectuate în vederea determinării valorilor statistice să fie egal pentru toate clasele.

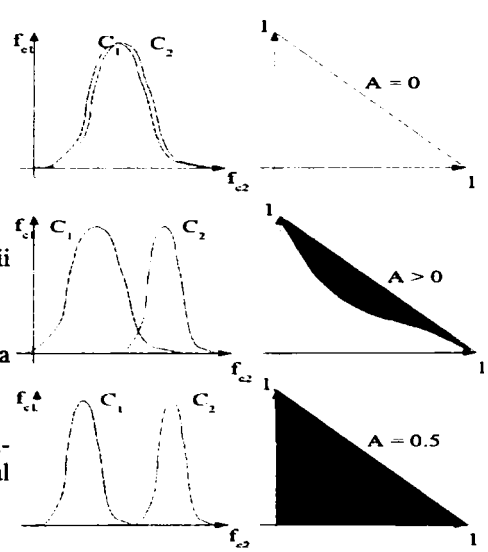


Fig. 197: Metoda ROC

Metoda ROC (Receiver Operating Characteristic)

Metoda se bazează pe analiza modului de suprapunere a distribuţiilor, măsurând suprafaţă normalizată a gradului de suprapunere a acestora. Schematic, acest lucru este prezentat în Fig. 197.

Aşadar, pentru o separabilitate optimală, valoarea rezultată este de 0.5.

5.2.5.3 Transformarea după axele principale

În cazul acestui tip de transformare, vor fi modificate axele sistemului de coordonate astfel încât să poată fi înlăturate corelaţiile dintre caracteristici. Astfel, axele sistemului vor fi rotite consecutiv până vor deveni paralele cu direcţiile corespunzătoare celor mai mari distribuţii. Se vor obţine astfel noi caracteristici statistice independente.

5.2.6 Clasificarea obiectelor

Utilizând metodele şi transformările prezentate anterior, se poate trece la realizarea ţelului dorit, de determinarea a hiperplanelor optime de separare, prin utilizarea metodelor de optimizare liniară sau neliniară, precum şi a noilor sisteme bazate pe algoritmi genetici sau reţele neuronale sau chiar a sistemelor de învăţare nesupraveghiată.

6 Ridicarea în 3D

Deşi la ora actuală nu există sisteme de ridicare în 3D de tip OpenSource, totuşi ţinând cont de faptul că modelele matematice există, vom aminti în continuare doar câteva lucrări publicate în care aceste modele şi algoritmii necesari au fost prezentate. Unele din motivele principale datorită cărora ridicarea în 3D este încă în faza incipientă sunt caracterizarea şi clasificarea. Prin dezvoltarea în continuare a algoritmilor necesari acestor modele precum şi a celor aferenta legate de domeniul de CV, se va ajunge cu siguranţă într-un viitor apropiat la elaborarea şi de programe OpenSource de acest tip.

6.1 Putting Objects în Perspective[47]/Automatic Photo Pop-Up[48]

Unul dintre cele mai impozante articole publicate în 2006 este cel cunoscut sub acest nume. Programul demostrativ elaborat, demonstrează posibilitatea ridicării în 3D a scenelor obţinute dintr-o singură imagine digitală. Exemplul prezentat în cadrul lucrării este redat în figura următoare:

Fig. 198: Automatic Photo Pop-Up

Pe lângă faptul ca sistemul necesită putere de calcul relativ mare (pentru o imagin 800x600 fiind necesare 1,5 Minute pe un calculator dotat cu Athlon 2,13 Mhz), instalarea unei versiuni runtime a pachetului MATLAB este de asemenea necesară. Utilizarea programului este gratuită, însa sursele nu sunt date publicităţii, fapt pentru care o adaptare a sistemului în vederea utilizării sale în scopul prototipării rapide este la ora actuală înca limitat şi doar interactiv posibilă.

6.2 DAVID[49]

Deşi la ora actuală există numeroase programe de scanare 3D, unul dintre cele mai interesante în special datorită următoarelor caracterisitci:

- preţul de achizi ie (gr t it s st m b ă ş sub 100€ pentru modulul 'Shapefusion' de fu- zionare a paşilor intermediari de scanare)
- nu necesită componente hard specifice sau specializate
- utilizarea unei mese rotative nu este obligatorie
- laserul utilizat este manipulat manual, permiţând baleierea zonelor problematice într-un raster mai dens
- export al obiectelor scanate în OBJ, PLY sau STL

Fig. 199: DAVID

7 Vizualizarea prin obiecte

Unul dintre cei mai importanți pași ai prototipării rapide virtuale este vizualizarea, prin care, datorită simulării prelucrării, se reduce simțitor timpul necesar obținerii obiectului final, erorile de prelucrare putând fi eliminate înaintea prelucrării reale a piesei. În prezent, vizualizarea este efectuată prin proiecții 2D a obiectului real, deși vizualizări de tip 3D (hologame, CAVE, Virtual Reality, Immersive Systems) sunt de asemenea existente în produse comerciale profesionale.

Din acest motiv, vom trata foarte pe scurt modelele și metodele matematice necesare vizualizării prin metode orientate pe obiecte.

7.1 Elemente de geometrie

Datorită faptului că în analiza și vizualizarea obiectelor simulate intervin patru sisteme de lucru, vom avea și patru sisteme de coordonate, pentru:

- model – sistemul de coordonate în care modelul este definit – în general un sistem de coordonate cartezian local 3D
- mediu (world) – sistemul de coordonate 'global' în care sunt poziționați atât 'actorii' (instanțe ale modelelor) cât și camerele și sursele de lumină
- view – sistemul de coordonate prin care este reprezentat tot ceea ce este vizibil printr-o anumită cameră (valorile axelor fiind cuprinse între -1 și 1)
- display – asemănător cu sistemulde coordonate utilizate de view, doar că valorile axelor de coordonate vor fi mapate pe pixeli.

Schematic, dependența acestor sistem este prezentată în figura următoare.

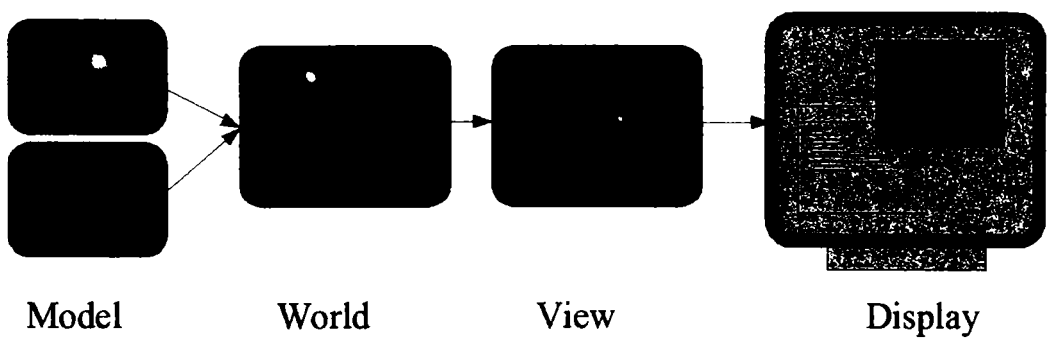


Fig. 200: Sisteme de coordonate

7.1.1 Transformări ale sistemelor de coordonate

Dat fiind faptul că obiectele definite în 3D trebuie proiectate pe planul 2D al imaginii (display) şi includ-ând efecte de proiecţe cum ar fi dispariţia puntelor şi obiectelor îndepărtate, este utilizat un sistem de coordonate omogen. Acest sistem de coordonate, diferă de cel 3D 'normal' prin faptul că utilizează în locul vectorului tridimensional de tip (x, y, z) prin elemente vectoriale de tip (x_h, y_h, z_h, w_h). Conversia dintre aceste două sisteme de coordonate este:

$$x=\frac{x_h}{w_h}, y=\frac{y_h}{w_h}, z=\frac{z_h}{w_h}$$

După cum se observă, într-un sistem de coordonate omogen, un punct aflat la infinit poate fi obţinut pentru valorarea $w_h=0,$ proprietate foarte importantă pentru transformările perspective ale camerei. Din acest motiv, transformările între sistemele de coordonate vor utiliza matrici de transformare de tip 4×4.

În continuare, vom prezenta pe scurt doar ecuaţiile necesare transformărilor sistemelor de coordonate 3D (2D fiind doar un caz particular al acestora).

Translatare

Translaţia unui punct în sistemul omogen de coordonate este dată de:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotaţie

Rotaţia putând avea loc după toate cele trei axe de coordonate, ecuaţia de transformare este:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\Theta_{x'x} & \cos\Theta_{x'y} & \cos\Theta_{x'z} & t_x \\ \cos\Theta_{y'x} & \cos\Theta_{y'y} & \cos\Theta_{y'z} & t_y \\ \cos\Theta_{z'x} & \cos\Theta_{z'y} & \cos\Theta_{z'z} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Scalare

Scalarea este transformarea dată de:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

7.1.2 Grafica asistată

În cazul vizualizării, elementul determinant este partea de grafică asistată. Prin aceasta, obiectele modelate

pot fi în mod corespunzător transformate pentru a obţine o imagine asemănătoarea uneia reale. La modul general, se poate spune că vizualizarea este procesul de transformare a datelor într-un set de obiecte grafice elementare, pe când grafica asistată transformă aceste elemente în imagini sau animaţii.

Din acest motiv, procesele şi metodele principale care stau la baza graficii asistate – şi deci având de asemenea un rol foarte important în ultimii paşi ai prototipării rapide – vor fi amintite pe scurt în cele ce urmează.

7.1.2.1 Redering - Descrierea fizică

Principalul element al graficii asistate este cel dat de rolul acesteia, deci de a genera imagini pentru diferite periferii din datele conţinute în obiecte elementare, proces care poartă denumirea de Rendering.

În funcţie de modelul funcţional abordat, exista numeroase metode utilizate în acest scop. Astfel, în cazul modelelor bazate pe faptul că razele provenite de la o sursă de lumină vor fi reflectate de obiectele reale (sau modelate), iar dintre acestea o parte vor excita celulele retinei ochiului uman sau luminoforii ecranului, se utilizează denumirea de ray-tracing sau ray-casting şi sunt cele mai utlizate. Există însă metode bazate pe algoritmi ce iau în considerare şi caracteristicile volumetrice ale obiectelor precum şi proprietăţile fizice ale acestora (e.g. radiosity, rendering volumic).

În vederea reducerii numărului de raze calculate, imaginea este calculată în mod invers, deci drumul parcurs de rază este străbătut de la punctul de analizat spre sursa de lumină.

7.1.2.2 Categorii de rendering

În funcţie de modul de generare a imaginii, procesul de este împărţit în două categorii:

- image-order – deci ordonat după imagini – specific metodelor de ray-tracing Într-un astfel de model, în general calculul începe cu primul pixel partea stângă superioară a ecranului şi este terminat cu ultimul pixel aflat în dreapta jos. Modelul este evident utilizabil doar pentru sisteme nevectoriale, cum ar fi plăcile grafice ale calculatoarelor sau imprimantele laser sau cu jet, dar destul de ineficiente pentru sisteme vectoriale cum ar fi cazul ploterelor.
- object-order – deci ordonat după obiecte – evident utilizabil doar pentru sisteme vectoriale (plotere în general, dar eventual şi pentru comanda maşinilor cu comandă numerică)

Pe de altă parte, în funcţie de modul de analizare a obiectelor de prelucrat, distingem:

- rendering de suprafaţă – în cazul căruia doar modul în care suprafeţele obiectelor interacţionează cu razele de lumină (gradul şi modul de reflexie)
- rendering volumic – în special pentru elemente transparente sau semitransparente, ceaţă, fum, nori, etc., precum şi în vederea generării de imagini multistrat (CT, MR – deci prin care interiorul corpurilor de asemenea este luat în considerare)

Evident că timpul de calcul necesar generării imaginii este proporţional cu complexitatea imaginii pe de-o parte, iar pe de altă parte de complexitatea categoriei de rendering utilizate şi a tipurilor de materiale întâlnite în cadrul scenei.

În ultimul timp, datorită apariţiei plăcilor grafice de performanţe ridicate, sistemele, metodele şi algoritmii de rendering au devenit tot mai perfecţionaţi, rezultatele procesului fiind de multe ori foarte greu de deosebit de imaginile lumii reale.

În ceea ce priveşte prototiparea rapidă, metodele de rendering de tip object-order sunt perfect utilizabile în vederea generării traiectoriilor sculelor, iar cele de image-order în vederea vizualizării rezultatului. Din acest motiv, prototiparea rapidă este unul dintre puţinele domenii în care în cadrul aceluiaşi sistem pot fi utilizate două metode diferite de rendering. De asemenea, prin modificarea proprietăţii de transparenţă a materialelor folosite, metodele de rendering volumic pot fi utilizate în vederea vizualizării obiectelor asamblate sau multimaterial, indiferent de metoda de vizualizare (display, imprimantă, Virtual Reality sau

Immersive Systems).

Unul dintre cele mai puternice programe de grafică existente la ora actuală şi care are marele avantaj că este de tip Open Source, este Blender. Programul permite programarea interactivă a scenelor, iar prin posibilităţile sale de cinematică inversă şi animaţie ar putea fi realizate simulări de prelucrare de calitate deosebită. La ora actuală programul este utilizat în special pentru efecte speciale în cinematografie, animaţii şi reclame dar prin faptul că utilizarea sa este gratuită, câştigă tot mai mult teren în tot mai multe domenii legate de grafica asistată (programarea de jocuri, artă digitală, etc.) motiv pentru care, primii paşi în vederea utilizării sale în vederea simulării fazelor procesului de prototipare rapidă fiind realizaţi în cadrul lucrării de faţă.

8 Sisteme de operare

În vederea prototipării rapide, unul dintre rolurile esențiale îl joacă sistemele de operare utilizate. În general, pînă acum câțiva ani, sistemele de operare ale mașinilor cu comandă numerică erau sisteme proprietare în timp real, foarte specializate, cu interfață umană relativ simplă, redusă la un panou de comandă manuală a mașinii, un monitor de urmărire, iar introducerea programelor era efectuată prin benzi magnetice, casete, dischete, cartele sau benzi găurite. Sistemele aferente de prelucrare a datelor provenite din programe CAD au fost cu timpul trecute de la sisteme de calcul de putere ridicată pe sisteme de tip personal, datorită dezvoltării explozive a microprocesoarelor și a calculatoarelor personale (PC).

Cu toate acestea, chiar și la ora actuală, utilizarea PC-urilor este în general redusă la programe CAD și partea de vizualizare, datorită modului de lucru a sistemelor de operare – în special a faptului că aceste sisteme nu au fost concepute de a lucra în timp real. Din această cauză, viteza de reacția a sistelului la evenimentele apărute și anunțate de senzori este redusă, ducând de cele mai multe ori la distrugerea sculei și chiar a mașinii.

În ultimii ani însă, dat fiind faptul că procesoarele, coprocesoarele și elementele electronice din componența calculatoarelor contemporane sunt de calitate și viteză de calcul mult mai ridicate decât a celor utilizate în anii anteriori, dar și în special a diferenței de preț ridicate dintre sistemele proprietare și PC-uri, a apărut necesitatea tot mai stringentă a implementării de sisteme de operare, standarde de prelucrare și metode de lucru adaptabile pe noile sisteme de calcul.

Astfel, posibilitatea de a genera programe CNC (în general în G-Code) este specifică majorității programelor CAD/CAM existente la ora actuală pe piață. Programe de cercetare a numeroase institute de cercetare și universități colaborează în vederea elaborării unui nou standard de prelucrare NC și CNC – ISO14649. Dintre aceste programe pot fi cu siguranță amintite STEP-NC în Europa și Asia, pecum și Super Model în USA.

8.1 Noi standarde

Motivul principal pentru care atât industria cât și învățământul colaborează la elaborarea noului standard, este faptul că cel vechi (ISO6893) elaborat de MIT la începutul anilor '50 este singurul standard care la ora actuală este utilizat cu succes în programarea mașinilor cu comandă numerică, dar care, datorită dezvoltării sistelemlor de calcul, a electronicii mașinilor cu comandă numerică și posibilităților acestora de interacționare (multi-axă, multi-scule și multi-proces), a devenit un sistem învechit, optimizabil.

Aceste dezvoltări au atras cu sine introducerea unui nou standard (ISO6983) prin care traiectoria sculei și starea mașinii pot fi de asemenea programate, însă limbajul de lucru este încă tot bazat pe vechiul sistem de comenzi G și M.

Ca urmare a faptului că încercări de elaborare de noi limbaje (SET, VDA sau IGES) deși aplicabile în anumite domenii, n-au putut satisface toate necesitătile impuse de industrie, comunitatea internațională a elaborat un nou set de standarde, ISO10303, cunoscut sub denumirea de STEP.

Avantajul esențial al sistemelor actuale, este în primul rând că, datorită puterii ridicate de calcul, o simulare a procesului de producție este de asemenea posibilă, ducând astfel la o reducere radicală a costurilor. Pe de altă parte, prin utilizarea sistemelor, metodelor și libajelor vechi, în momentul transferării datelor de la un sistem CAD la altul, sau CAD-CAM, numeroase informații se pierd.

Tendința actuală,cea de utilizare a arhitecturilor deschise (open architecture - OSACA, OMAC) atrage cu sine necesitatea elaborării atât a unui limbaj comun CAD, CAM, CAPP și CNC, cât și a unei abstractizări a sistemelor hardware.

Limbajul care la ora actuală pretinde să realizeze cel mai mult din acest deziderat poată denumirea de STEP-C-NC (STEP Compliant NC) și a fost standardizat, după cum am amintit, sub codul ISO14649.

Principial, acest standard poate fi reprezentat ca în Fig. 204:

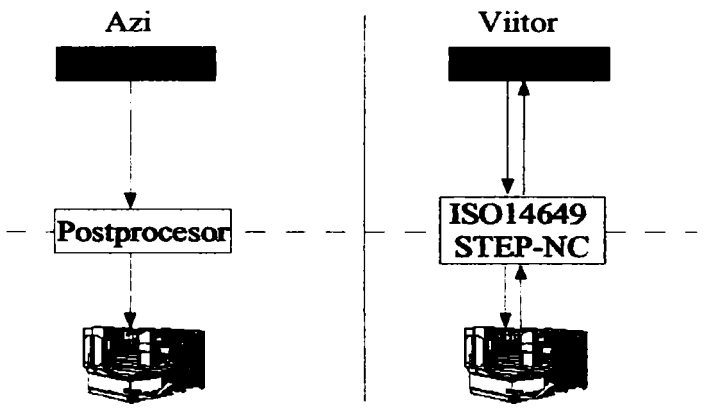


Fig. 201: ISO 14649 - principiu

Acest standard, nu este nici un limbaj sau metodă de programare şi nici nu descrie traseul sculei de prelucrare, ci este un model obiect orientat pentru CNC în care datele sunt structurate şi detaliate printr-o interfaţă bazată pe atribute (feature based).

Principial, o astfel de interfaţă este prezentată în Fig. 205 ([50]):

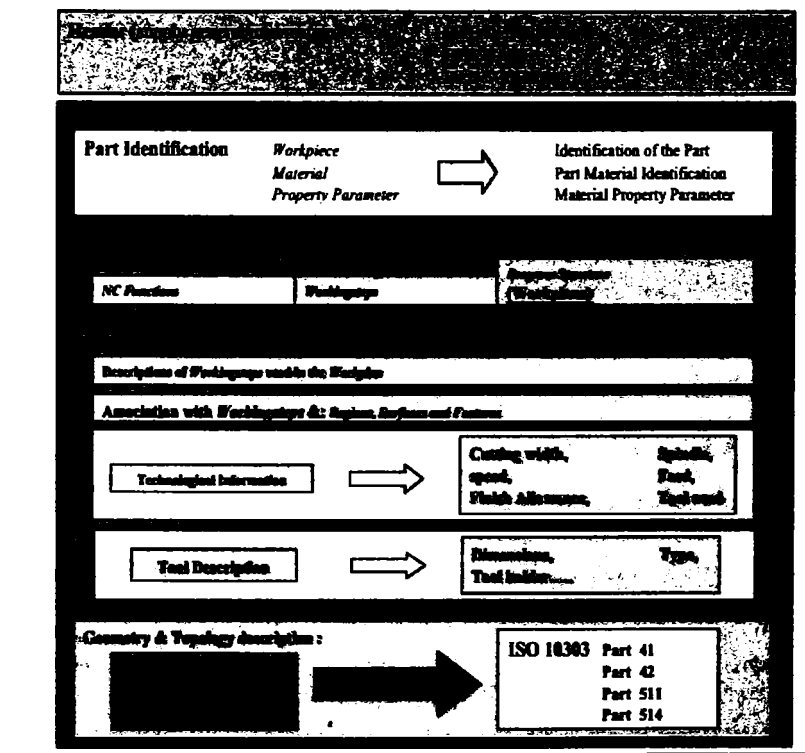


Fig. 202: STEP-C-NC

Un exemplu, prezentat în aceeași sursă are forma:

```
HEADER;
FILE_DESCRIPTION(('Example of NC programme fo milling: Planar Face, Pocket,Hole '),'1');
FILE_NAME('example1 .stp',$,('ISO14649'),(''),'Jochen Wolf','WZL RWTH-Aachen','Germany');
FILE_SCHEMA(('MACHINING_SCHEMA','MILLING_SCHEMA'));
ENDSEC;
DATA;
#1=WORKPIECE('simple workpiece',#2,0.010,$,$,$,(#57,#58,#59,#60));
#2=MATERIAL('ABNT1045','Steel',(#3));
...
#5=ROUND_HOLE('Hole1 D=22mm',#1,(#19,#20),#43,#63,$,$,$,$,#64,$,#6,$,$);
...
#9=PROJECT('Execute example1',#10,(#1));
#10=WORKPLAN('Main workplan',(#11,#12,#13,#14,#15),$,#36);
#11=MACHINING_WORKINGSTEP('WS finish Planar Face1',#46,#4,#16);
#12=MACHINING_WORKINGSTEP('WS drill Hole1',#46,#5,#19);
#13=MACHINING_WORKINGSTEP('WS Ream Hole1',#46,#5,#20);
#14=MACHINING_WORKINGSTEP('WS rough Pocket1',#47,#7,#21);
#15=MACHINING_WORKINGSTEP('WS finish Pocket1',#47,#7,#22);
...
#19=DRILLING($,$,'drill Hole1',#26,10.000,$,#48,#31,#35,$,0.000);
...
#26=DRILLING_TYPE_STRATEGY(75.000,50.000,2.000,50.000,75.000,8.000,10.000,$,200.000);
...
#31=MILLING_TECHNOLOGY(0.0300,.TCP.,$,1.600,$,$,F,F,F);
...
#48=MILLING_CUTTING_TOOL('Spiral_drill_20mm',#51,(),0,90.000,$,$);
...
#51=TAPERED_DRILL(#54,2,$,F,$,30.000);
...
#54=TOOL_DIMENSION(20.000,59.000,$,$,$,$,$);
...
#63=TOLERANCED_LENGTH_MEASURE(30.000,$,1.000,1.000);
...
#86=CARTESIAN_POINT('Hole1: Location ',(20.000,60.000,45.000));
...
ENDSEC;
```

Fig. 203: STEP-C-NC exemplu

Multe din operațiile cunoscute (sau chiar faze de prelucrare) sunt incluse în acest standard, dat fiind faptul că în general operațiile au loc în același mod, diferiți fiind doar parametrii utilizați pentru G și M. Este deci esențial, ca sistemele CAD/CAM să utilizeze cazul cel mai general din cadrul standardului, în vederea evitării unei simple noi parametrizări a programelor CNC existente și deci ducând la 'pierderea' nivelului de inteligență a operării. În momentul de față, dat fiind faptul că foarte puține mașini de prelucrare CNC utilizează acest sistem, pierderea n-ar fi deosebit de gravă. Este însa esențial pentru generațiile viitoare de mașini-unelte dotate cu inteligență artificială să le lăsăm libertatea alegerii traiectoriilor sculelor și eventul chiar a ordinii prelucrărilor. Chiar dacă adaptarea automată a unelor caracteristici de prelucrare aparține unui viitor mai îndepărtat sau eventual chiar imposibilă (de exemplu adaptarea prelucrărilor în 5 axe pe o mașină cu 3 axe), este totuși indicată utilizarea cât mai exactă a standardului amintit, pentru că viitorul începe cu siguranță acum.

Principalele avantaje utilizării unui astfel de sistem sunt deci:

- limbaj unic pentru CAD, CAM, CAPP și NC
- programul NC va putea fi generat în viitor în mod automat, dat fiind faptul că planificarea detaliată a procesului este de asemenea conținută în datele de intrare

8.2 Sisteme de operare în timp real

După cum am amintit anterior, dezavantajul esențial al sistemelor de calcul de tip PC în vederea prelucrărilor CNC este operarea în timp real (RT). Principalele sisteme de operare utilizate la ora actuală pe PC-uri, Windows și Linux, nu sunt capabile de a opera în acest mod. Din această cauză, aceste sisteme sunt utilizate aproape în mod exclusiv pentru sistemele CAD și în vederea vizualizării prelucrărilor.

Pe de o parte, rolul sistemelor de tip PC în vederea designului, simulării și vizualizării a crescut proporțional cu scăderea prețului lor. Pe de altă parte, sistemele proprietare de automatizare au prețuri ridicate în ciuda faptului că majoritatea componentelor (motoare pas cu pas, servomotoare cu comandă digitală, etc.) au devenit tot mai accesibile. Aceste considerente, la care vom mai adăuga faptul că există variante de operare în timp real a sistemului Linux (RTLinux – e.g. BDI) a dus la apariția primelor mașini cu comandă numerică a căror operare este în întregime efectuată pe PC.

Pe lângă faptul că este necesar, ca sistemul să opereze în timp real, a fost necesară programarea unui sistem de control CNC capabil să prelucreze datele de intrare și ieșire necesare prelucrării mecanice. Astfel, în anul 2002 a apărut prima versiune a programului EMC (Enhanced Machine Controller), und controler CNC de tip freeware și open source, sub egida NIST (National Institute for Standards and Technology).

Programul și-a propus operarea în timp real prin intermediul portului paralel al PC-ului a motorarelor pas cu pas, a servomotoarelor, releelor precum și a contactelor în vederea programării și controlării roboților, a mașinilor cu comandă numerică sau a altor aparate din domeniul automatizării.

Astfel EMC poate la ora actuală comanda motoare pas cu pas prin trimiterea de impulsuri prin pinii portului paralel, sau prin comandarea plăcilor de interfață a servomotoarelor.

De asemenea, programul în G-Code poate fi transmis interfeței de comandă (MDI – Machine Device Interface) în vederea prelucrării sau, bineânțeles, ca fișier.

Interfața grafică a sistemului este foarte variată - text, Xwindow, Tcl/Tk, Java ...

În general, portul paralel al PC-ului este utilizat în modul următor:

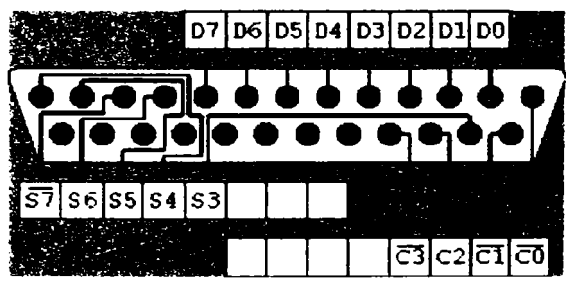


Fig. 204: EMC – conectarea portului paralel

Se vede astfel, ca prin utilizarea acestui port cu motoare pas cu pas binare, vom putea controla concomitent până la 6 motoare (12 biti de ieșire). În schimb, dacă dorim ca poziționarea după axele de coordonate în domeniul maxim pozitiv, negativ și HOME să fie de asemenea luată în considerare, vom observa că doar o singură axă poate fi complet integrată (pentru fiecare axă fiind necesare 3 semnale de intrare – interfața având doar 5). Soluția este de a instala încă unul sau două porturi paralele, în acest caz având posibilitatea utilizării mai multor parametri de intrare și ieșire, sau a cuplării limitelor celor 3 axe.

În cazul a două portul paralele, modul de atribuire a conexiunilor este prezentat în Fig. 205.

IO Pin	Function
D0, pin 2	X direction
D1, pin 3	X clock
D2, pin 4	Y direction
D3, pin 5	Y clock
D4, pin 6	Z direction
D5, pin 7	Z clock
D6, pin 8	A direction
D7, pin 9	A clock
C0, pin 1	B direction
C1, pin 14	B clock
C2, pin 16	C direction
C3, pin 17	C clock
S3, pin 15	X/Y/Z lim +
S4, pin 13	X/Y/Z lim -
S5, pin 12	X/Y/Z home
S6, pin 11	Probe
S7, pin 10	spare

IO Pin	Function
D0, pin 2	Spindle reverse
D1, pin 3	Spindle Forward
D2, pin 4	spare
D3, pin 5	Spindle on
D4, pin 6	spare
D5, pin 7	spare
D6, pin 8	Mist Coolant
D7, pin 9	Flood coolant
C0, pin 1	Speed decrease
C1, pin 14	Speed increase
C2, pin 16	Estop output
C3, pin 17	Spindle brake
S3, pin 15	spare
S4, pin 13	Estop input
S5, pin 12	Lube input
S6, pin 11	spare
S7, pin 10	Spare

Fig. 205: EMC - conexiuni pentru 2 porturi paralele

Unele porturi paralele permit chiar programarea direcţiei datelor, astfel încât portul paralel să utilizeze 8 pini de ieşire şi 9 de intrare, suficienţi pentru controlul mişcărilor după 3 axe.

Prin utilizarea a 3 porturi paralele, vom avea 36 de pini de ieşire şi 15 de intrare. Partea de controler a motorului pas cu pas este implementată într-un thread RT secundar cu o rată de tastare de 100µs (deci pentru un impus sunt necesare 200µs respectiv o frecvenţă de 5 kHz).

Datorită faptului că acest program este open source, prin contribuţia multor programatori, în anul 2005 a fost eliberată o nouă versiune – EMC2.

8.3 Abstractizarea sistemelor hard (HAL – Hardware abstraction layer)

Pornind de la simpla constatare că toate maşinile-unelte existente la ora actuală, în cazul în care ele ar fi asamblate de o singură persoană, această persoană nu va trebui în mod necesar să cunoască modul de funcţionare şi construcţie a fiecărei părţi componente (motoare sau servomotoare, schimbătoarele de piese, contactoare, comutatoare, codificatoare, etc.), ci doar modul în care acestea trebuie să fie conectate în vederea obţinerii maşinii dorite (principiul de black box). Este deci posibilă definirea unui sistem hardware specific abstractizat, în care doar interfeţele componentelor sunt definite, dat fiind faptul că în toate sistemele cunoscute, interfeţele sunt bazate pe semnale de intrare şi ieşire, modul de intern funcţionare fiind irelevant pentru funcţionarea sistemului. Constructorul sistemului nou, are astfel singura sarcină de a conecta intrările şi ieşirile părţilor componente în mod corespunzător cerinţelor sistemului ce urmează a fi realizat.

Sistemul prezentat în continuare, deşi doar în februarie 2005 documentat, a fost deja incorporat nu numai

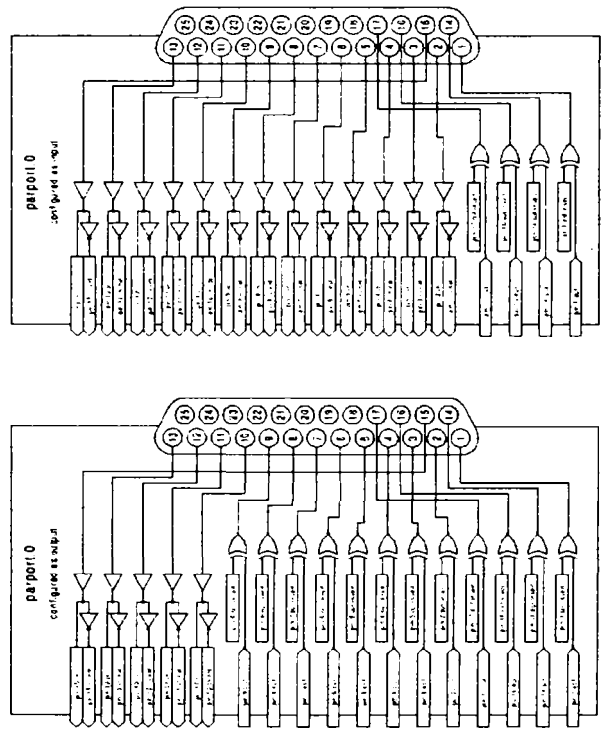


Fig. 206: HAL - reprezentarea portului paralel

în noul sistem prezentat anterior (EMC2) ci şi în majoritatea distribuţiilor Linux actuale existente. Un exemplu de abstractizare a portului paralel de legătură cu maşina de prelucrare prin comandă numerică este prezentat în figura anterioară.

8.4 Sisteme distribuite (GRID-Computing)

Datorită faptului că mulţi din algoritmii prezentaţi nu pot fi rulaţi decât în cazuri particulare sau simple pe un singur calculator, dată fiind capacitatea de calcul ridicată necesară, într-un viitor mai mult sau mai puţin apropiat utilizarea de tehnologiilor GRID nu va putea fi ignorată. Majoritatea simulaţiilor la ora actuală (deşi încă în fază incipientă) are loc utilizând tehnologii de aplicaţii distribuite. Dacă ţinem cont de exemplul prezentat în cadrul recunoaşterii formelor simple (linii şi cercuri) şi al caracterizării proprietăţilor, unde numărul de variante de calculat este de ordinul $1{,}7 \cdot 10^{41}$, devine evident faptul că obţinerea rezultatelor pe un singur calculator este în cel mai bun caz neoptimală - cel puţin din punct de vedere a timpului necesar.

Din acest motiv, una dintre opţiunile posibile este paralelizarea calculelor şi distribuirea lor pe diferite sisteme. De menţionat este faptul, că deşi această tehnologie este utilizată în momentul de faţă în special în domeniul HPC, tehnologia este de tip OpenSource şi este utilizabilă fără restrângeri pe orice tip de calculatoare - bineînţeles cu condiţia ca acestea să fie conectate într-o reţea accesibilă, iar programele necesare

să fie instalate pe sistemele apelate. Astfel, utilizarea rețelei interne a unei universități, a unui centru de calcul, precum și a mai multor calculatoare private este mai mult decât adecvată în vederea obținerii rezultatelor dorite.

Scopul principal, și derivat de la acesta. folosul principal al utilizării acestei tehnologii în cazul nostru, este utilizarea rațională și cât mai optimă a resurselor de calcul existente. Prin această tehnologie este posibilă utilizarea resurselor de calcul existente în perioade de timp în care necesitatea medie de calcul este redusă (noaptea, pauze, etc.) în scopul obținerii mult mai rapide a rezultatelor calculelor intensive.

8.4.1 Projecte GRID existente

În vedera demonstrării potențialului existent în spatele noțiunii de GRID-Computing și fără a diminua importanța celorlalte proiecte (de exemplu legate de studiul și cercetarea climei și atmosferice, a textelor și beletristicii, a fizicii teoretice și aplicate, medicinei, chimiei, astronomiei, etc.), vom prezenta pe scurt cele ale inițiativei D-GRID[51] legate de inginerie sau tangențial acesteia. De menționat este faptul că, deși domeniul tehnic este prezent - după cum vom vedea în numeroase proiecte - în zona caracteristică prototipării rapide nu există nici unul, în ciuda posibilităților remarcabile existente.

8.4.1.1 AeroGrid

Proiect de colaborare bazat pe GRID între industrie, institute de cercetare și universități în domeniu aeronautic.

Scopul proiectului AeroGrid este de a pune la dispoziție un mediu de lucru bazat pe tehnologii GRID eficient pentru cercetarea aeronautică germană. Acest mediu de lucru va deveni o infrastructura Grid persistentă și orientată practic, în vederea cooperării între cercetare, industrie și universități în domeniul aeronautic. Comparativ cu alte sectoare industriale, o foarte mare parte a cercetării din acest domeniu are loc în instituții și organizații publice. Astfel, în domeniul simulării numerice a fluidelor, industria utilizează părți de cod cu algoritmi inovativi în mod direct, precum și părți de cod dezvoltate și implementate în cooperare între institute de cercetare - ca de exemplu DLR și universități.

Mediul de lucru va asigura o cooperare flexibilă în organizații virtuale în funcție de necesitățile concrete de project, precum și utilizarea independentă de zona geografică a tuturor versiunilor de program, a datelor și resurselor de calcul precum și o deductibilitate detaliată a modului în care rezultatele au fost determinate.

Concepția de pricipală a proiectului este bazată pe două cerințe primare:

- Aplicabilitatea practică, în special în vederea trecerii în producție după finalizarea proiectului printr-un Service-Provider.
- Portabilitatea pentru alte instituții și grupări asemănătoare din cercetare și industrie.

8.4.1.2 F&L-Grid – Forschung und Lehre Grid

În cadrul proiectului F&L-Grid va fi elaborat un Service-Grid generic în scopul ofertei de servicii IT de tip Public Private Partnership (PPP) în cadrul rețelei de cercetare germane (DFN).

Este urmărită generarea unei platforme GRID în vederea expandării serviciilor oferite de DFN. Aceste servicii vor fi implementate în cadrul proiectului de către firmele T-Systems și Karlsruhe Institute of Technology (KIT), conceptul fiind deschis pentru extinderi ulterioare. Universitatea Marburg colaborează de asemenea prin dezvoltări proprii de utilitare la integrarea de sistem. Pilotarea proiectului este organizată de către DFN, care intenționează preluarea în urma unei oferte de servicii ulterioare perioadei de subvenționare. Prin implicarea DFN și KIT în proiect, este asigurată totodată și o legatură strânsă a activităților cu cele din cadrul proiectului de integrare a serviciilor de GRID.

Cu toate că analiști consacrați (e.g. Insight Group) consideră Service-Grids ca fiind importante din punct de vedere economic doar într-un termen mai îndepărtat, acest proiect va deveni chiar în această fază atractiv din acest punct de vedere. Proiectul propus aici este de tip Path-Forward utilizând atât Enterprise- și

Utility-Grids existente cât şi cele de tip Open Service-Grid sau Service Oriented Knowledge Utility (SOKU). Din acest motiv, pentru a diminua riscurile ce apar în urma acestor integrări, următoarele premize vor fi necesare:

1. Cadrul de utilizatori nu va fi complet deschis, ci pentru început doar pentru anumite tipuri de utilizări ale reţelei de cercetare clar definite şi organizate. O deschidere ulterioară în direcţia intreprinderilor economice va fi analizată în timpul dezvoltării proiectului.

2. Serviciile oferite, vor fi în primul rând restricţionate la teme a căror controlabilitate într-un astfel de scenariu este evidentă.

Modul ales de lucru este astfel progresiv referitor la modelele de afaceri (bussiness models), dar conservativ referitor la tehnologiile aplicate.

8.4.1.3 InGrid - Innovative Grid-Entwicklungen für ingenieurwissenschaftliche Anwendungen

Proiectul comunitar InGrid permite atât utilizarea eficientă a aplicaţiilor de tip GRID precum şi cea a resurselor ce calcul (hard şi soft) în cadrul proiectelor de inginerie, prin implementarea unei platforme GRID specifice acestui domeniu. Prin utilizarea flexibilă a tehnologiilor GRID sunt compactate nu numai competenţele în domeniul modelării, simulării şi optimizării, ci totodată este obţinută o eficienţă ridicată a utilizării resurselor existente.

Cinci domenii tipice de aplicaţie vor fi în mod exemplar dezvoltate – turnarea, deformarea plastică, mecanica fluidelor, simularea turbinelor şi a interacţiunii dintre curenţi şi structurile mecanice – în vederea acoperirii a trei domenii centrale ale ingineriei pregnate de calcule intensive – probleme de scalare multiplă, probleme multidisciplinare şi optimizare simulată distribuită. În principal vor fi dezvoltate modele de proces scalabile şi platforme de rulare bazate pe tehnologii GRID în vederea rezolvării problemelor amintite anterior.

Proiectul este pozitionat atât în zona cercetării fundamentale cât şi a cercetării aplicate în vederea stabilizării unei infrastructuri de soft GRID specifice activităţilor inginereşti. Rezultatele vor fi utilizate pe trei nivele, cu scopul îmbunătăţirii interactivităţii dintre cercetare, dezvoltare şi producţie, în principal prin implementarea de componente soft pentru infrastructura eScience germană şi europeană.

Clasele de probleme cercetate sunt în momentul de faţă:

I. Generarea optimală de schiţe, planuri şi proiecte
II. Controlul optimal al sistemelelor
III. Designul sistemelor hibride
IV. Optimizarea sistemelor de producţie în mai multe trepte

Aceste probleme reprezintă părţi constituente ale prototipării virtuale şi a ingineriei colaborative. Doar prin utilizarea technologiilor GRID este asigurat accesul la resurse (procesor, memorie, utilitare, sisteme de informaţii combinat în mod adecvat şi adaptat cerinţelor specifice) cu reţele rapide şi sigure, pentru ca soluţiile rezolvabile în aceste domenii să ducă la o cooperare rentabilă între cercetători, dezvoltatori de programe, producători şi utilizatori. Astfel, graniţele clasice existente azi între cercetare şi dezvoltare specifice ingineriei şi economiei vor fi depăşite în favoarea unui evoluat ciclu de viaţă al produsului.

Ţelurile specifice comunităţii pot fi considerate din acest punct de vedere ca fiind:

1. Suport pentru mijloacele de prototipare virtuala
    a. metode colaborative Web-based
2. Reprezentarea de Workflows specifice ingineriei în mediul GRID
3. Ataşarea surselor specifice
4. Implementarea de Portale pentru scenarii specifice ingineriei
5. Dezvoltarea unui sistem de Knowledge-Management în vederea optimizării proceselor şi succesiunilor proceselor de decizie
6. Dezvoltarea de modele de cooperare, bussiness, securitate şi încredere specifice

9 Simularea

Prin faptul că utilizarea sa conduce la o reducere substanţială atât a timpului dezvoltării produsului cât şi a preţului acestuia, simularea este unul din elementele cele mai importante ale prototipării rapide. Metodele şi algoritmii necesari sunt într-o continuă perfecţionare, la ora actuală majoritatea testelor de crash fiind reduse la simulari numerice. În general însă, utilizarea programelor necesare simulării necesită investiţii ridicate (putere de calcul ridicată – HPC, GRID, sisteme imersibile de tip CAVE, etc.). În continuare vom încerca să demonstrăm că simularea simplă a prototipării rapide este posibilă şi prin utilizarea de utilitare OS, chiar dacă momentan demonstraţia este aplicabilă în special în scopuri didactice. Posibilităţile existente, în special datorate numărului imens de potenţiali programatori, vor conduce cu siguranţă la o dezvoltare accelerată a simulării în această direcţie.

9.1 Blender ca interfaţă pentru simularea NC

Idea principală a utilizării programului Blender ca interfaţă pentru simulare, am considerat-o ca o consecinţă firească a faptului că acest program permite dezvolarea interactivă de utilitare specifice CAD şi CAM printr-o interfaţare cu unul dintre cele mai puternince interpretoare existente – python. Pe de altă parte, primele încercări în direcţia roboticii, simulării curgerii fluidelor şi a simulării ciocnirilor elastice şi plastice au fost efectuate utilizând acest program.

9.1.1 Modelare

După cum ma mai amintit, unul dintre cele mai competitive utilitare de modelare de tip OS este Blender[52]. Facilităţile sale în ceea ce priveste modelarea şi vizualizarea sunt din multe puncte de vedere mai bune decât a celor mai multe produse de tip comercial comparabile existente (iar uneori mai eficiente decât acestea puse împreună). După cum este descris în [53] elementele principale caracteristice ale utilitarului sunt:

- Compendiu complet integrat de creaţie, oferind o gamă foarte largă de utilitare de bază inclusiv în vederea creerii elementelor de 3D, modelare, animaţie, rendering post-producţie video şi implementarea de jocuri
- executabil redus şi uşor de distribuit
- Architectura 3D de calitate ridicată, permiţând o modelare rapidă şi eficientă
- suport gratuit prin www.blender3d.org
- 250k+ utilizatori

Dat fiind faptul că programul poate fi obţinut în sursă, posibilitatea modificării codului şi adaptării sale în funcţie de necesităţile curente este de asemenea posibilă (bineînţeles cu condiţia respectării licenţelor soft incorporate).

Primii paşi în utilizarea programului Blender în domeniul mecanic au fost deja făcute de către Dept. of Mechanical Engineering K.U. Leuven, Belgium[54]. Ţinând cont de faptul că CNC şi robotica au foarte multe puncte comune, în special în ceea ce priveşte simularea şi vizualizarea, vom demonstra posibilitatea utilizării acestui program în vederea prototipării rapide.

Din acest motiv, utilizarea lui Blender se datorează în principal următoarelor caracteristici:

- un design de bază foarte bun având un cod separabil şi accesibil
- suport excepţional pentru rendering şi simularea mişcării
- bun suport pentru generarea de animaţii şi video
- foarte rapid
- adaptabil
- o "competiţie" încă imatură în robotică şi inexistentă în domeniul NC/CNC

- posibilitățile de plug-ins deschid o gamă foarte largă de posibilități (ODE și ILM prezentate în continuare fac deja parte din Blender)
- o foarte buna interfațare și integrare cu C++ și python

Din nefericire însă, Blender nu este nici pe departe un sistem adaptabil perfect, o parte din detrimente fiind cele prezentate în continuare:

- Blender lucrează în 'Blender units', pe când ingineria necesită unități fizice reale, ca de exemplu milimetri sau inch. Deci ceea ce este reprezentat în Blender trebuie sa *fie* realistice și nu doar să *pară* realistice
- Cinematica inversă (IK) este încă prea simplă dat fiind faptul că deocamdată doar un algoritm general este oferit
- Interpolarea (IPO) curbelor în Blender conține doar curbe spline și trepte între frames, pe când în ingineria avem nevoie de multiple familii de IPO în vederea calculării și generării de traiectorii

Utilizând însă facilitățile oferite de integrarea interpretorului python, posibilitâțile remarcabile ale lui Blender combinate cu flexibilitatea acestul limbaj modern oferă o platformă optimală pentru dezvoltarea rapidă a simulărilor, una dintre atributele cele mai importante ale RP.

Deși în Blender lipsesc capabilitățile CAD, dat fiind faptul că este în primul rând conceput pentru prezentări și animații, este cu siguranță posibilă simularea de dispozitive complicate și a operațiilor necesare cu un efort mult mai redus decât prin utilizarea altor programe comerciale existente. Exemplele prezentate în continuare sunt relativ simple, dar permit evidențierea posibilităților oferite de acest program și sper că vor deschide noi orizonturi în special în RP.

9.1.2 Simulare

Open Dynamics Engine (ODE) este o bibliotecă liberă de calitate industrială pentru simularea dinamicii articulațiilor rigide[55], utilizabilă pentru simularea mișcării vehiculelor, creaturilor cu picioare și mișcarea obiectelor în mediul VR. Este o bibliotecă de algoritmi rapizi, flexibili și robuști, având integrate module de detecția coliziunilor și este foarte stabilă în ciuda faptului că versiunea actuală este doar 0.9. Principalele atribute caracteristice sunt:

- bună simulare a articulațiilor rigide
- designed pentru a fi utilizat în simulări interactive sau în timp real. Utilizatorul are posibilitarea de a modifica structura mecanismului chiar și în timpul rulării simulării
- utilizează un stabilizator integrat care asigură faptul că erorile de simulare nu vor evolua în afara limitelor de control
- constrângeri de tip non-penetration sunt utilizate în momentul ciocnirii a două rigide
- are un sistem de detecție a coliziunilor integrat, dar care poate fi ignorat în cazul în care utilizatorul preferă un sistem de detecție propriu. Primitivele existente până în momentul de față sunt de tip box, sferă, cilindru, plan, rază și triunghi. De asemenea oferă o identificare rapidă a unei eventuale intersecții a obiectelor prin conceptul de "spații"
- este bazat pe fizica corpurilor rigide cu masă arbitra distribuită
- numeroase articulații implementate până în momentul de față
- metoda de simulare: Ecuațiile de mișcare sunt derivate dintr-un model de bază al multiplicatorului de viteză Lagrange dat de algoritmii Trinkle/Stewart și Anitescu/Potra
- pînă în momentul de față integrarea de ordinul unu a fost utilizată, deci deși rapidă, încă nu destul de exactă pentru inginerie. Integrări de ordin mai mare sunt în implementare
- două metode de deplasare în timp sunt posibile - metoda standard "big matrix" sau o nouă metodă interactivă numită "QuickStep"

- modelul de contact și frictiune este bazat pe algoritmul Dantzig LCP descris de Baraff, deși ODE implementează o metodă de aproximare mai rapidă bazată pe modelul de frictiune Coulomb
- interfață C (deși ODE este în cea mai mare parte scris în C++)
- interfața C++ este bazată pe cea scrisă în C
- multe teste unitare existente și multe încă în implementare
- optimizări specifice platformei actuale de lucru
- biblioteca este FreeSofltware și poate fi distribuită sub GNU Lesser General Public License sau BSD-Style

Fizica corpurilor rigide se bazează pe faptul că corpurile au proprietăți dinamice și constante în timp.

Cele dinamice sunt:

- poziția (pentru moment cea corespunzătoare centrului de masă) reprezentată de vectorul (x, y, z)
- viteza liniară (v_x, v_y, v_z)
- orientarea, reprezentată fie printr-o matrice de rotație de tip 3×3 sau printr-un quaternion (q_s, q_x, q_y, q_z)
- un vector unghiular de viteză descriind schimbarea de viteză în timp (w_x, w_y, w_z)

Proprietățile constante sunt:

- masa corpului rigid
- poziția relativă a centrului de masă față de punctul de referință (în versiunea actuala ele trebuie sa coincidă)
- matricea de inerție care descrie distribuția în jurul centrului de masă

Principial, fiecare corp rigid are un sistem de coordonate $x-y-z$ înglobat, care este translatat și rotit odată cu obiectul pe care-l reprezintă așa cum este prezentat în Fig. 207.

Fig. 207: Sistem local de coordonate

O altă caracteristică foarte importantă este cea a posibilității de activare/dezactivare a corpurilor în vederea participării sale la iterația următoare. Prin acest fapt, timpul necesar simulării poate fi considerabil redus în cazul în care anumite corpuri n-au nici o influiență asupra simulării sau sunt fixe. Dat fiind faptul că anumite corpuri rigide interacționează pe durata simulării, noțiunea de „insulă" a fost definită în scopul desemnării grupurilor de corpuri rigide care sunt inseparabile. Din acest motiv, o insulă este dezactivată doar dacă toate corpurile care o compun sunt la rândul lor dezactivate. În cazul în care cel puțin un corp este activat, toate celelalte vor lua parte la următoarea iterație.

Chiar dacă în momentul de față, datorită impedimentului amintit referitor la precizia de calcul, ODE nu este încă recomandabil pentru calcule specifice ingineriei și deci rezultate din iterații numeroase, integratorul este suficient de stabil pentru a fi folosit în RP, datorită numărului relativ redus de iterații necesare.

O altă proprietate foarte importantă este implementarea de articulații și a restricțiilor lor corespunzătoare. Astfel, pentru o articulație ca și cea prezentată în Fig. 208 un paramentru de corecție va asigura faptul ca cele două rigide componente să nu fie separate, deci articulația nu va putea niciodată apărea ca în Fig. 209.

Fig. 208: Exemplu de articulație cu ERP

Fig. 209: Exemplu de articulație fără ERP

Constraint Force Mixing (CFM) este o altă caracteristică implementată în ODE. Notând Jacobianul (conținând câte un rând pentru fiecare grad de libertate a sistemului) cu J, viteza cu v și considerând vectorul constant c, ecuația articulației va fi utilizând un algoritm tradițional:

$$J * v = c \tag{1}$$

Introducând însă un vector λ (de aceeași dimensiune ca și c) forța necesară pentru menținerea articulației va fi obținută prin:

$$f = J^T * \lambda \tag{2}$$

ODE introduce o nouă matrice diagonală patrată C_{FM} care îmbină forțele rezultate cu cele de restricție . astfel încât ecuația (1) devine:

$$J * v = c + C_{FM} * \lambda \tag{3}$$

Utilizând valori pozitive pentru C_{FM} sistemul nu mai devine singular iar precizia este îmbunătățită.

9.1.3 Vizualizare

După cum este descris pe Homepage ([56]) "OpenEXR este un format de fisier pentru imagini digitale de tip high dynamic-range (HDR), dezvoltat de Industrial Light & Magic[57] (ILM) pentru utilizare în aplicații de prelucrare a imaginilor digitale". Caracteristicile principale ale acestui produs sunt:

- spectru dinamic și precizie a culorilor mult mai mare decât a altor formate pentru imagini digitale
- suport pentru 16, 32 bit floating-point și 32 bit integer pixels (modul de 16 bit floating-point este compatibil cu "half" graphics language utilizat de NVIDIA (Cg) utilizând astfel în mod nativ cipurile GeForce FX și Quatro FX 3D
- algoritmi de compresie fără pierdere multiplă prin care se pot atinge grade de comprimare de pînă la 2:1
- extensibilitate ridicată și backward compatibility datorită implementării în C++
- portat pe GNU/Linux, OS X, Win32 și IRIX dar relativ simplu de portat pe orice alte sisteme de operare UNIX-like

ILM a lansat inițial OpenEXR ca free software, dar în momentul de față licența a fost modificată în BSD. Distribuția soft include:

- IlmImf – o bibliotecă de citire și scriere a fisierelor de imagini OpenEXR

- Half – o clasă C++ pentru manipularea de „half values" tratabile ca structuri C++
- Imath – biblioteca matematică pentru operații pe matrice, transformări 2D și 3D, rezolvari de ecuații liniare, patrate și cubice și multe, multe altele...
- exrdisplay – un exemplu simplu de aplicație pentru vizualizarea imaginilor OpenEXR cu diferiți parametri de setare

9.2 Simularea frezarii și gauririi

Deși evident simularea frezării este mai complicată decât cea a strunjirii, datorită faptului că în cadrul lucrării ne vom ocupa doar de urmărirea traiectoriilor generalizate modelate în Blender, cazul frezării este din acest motiv mult simplificat și ne vom ocupa în primul rând de acest tip de prelucrare.

De asemenea, ținând cont de faptul că din punctul de vedere al generării programelor CNC între 'dwelling' și găurire nu există diferențe, ambele utilizând o poziționare fixă pe două dintre axe și o deplasare liniară pe cea de-a treia, vom trata gaurirea pe mașini cu comandă numerică concomitent cu frezarea. Cu toate că sursele programelor implementate sunt de relativ reduse și cuprinse în anexate lucrării, cele G-Code generate sunt în general foarte mari, reprezentarea lor mărginindu-se din acest motiv doar la imaginile simulării sau prelucrării pe mașina corespunzătoare prin comandă numerică.

9.2.1 Frezarea/tăierea unui contur rezultat din analiza unei imagini digitale

În vederea simulării, rezultatul binar al analizei imaginilor digitale va fi utilizat ca fișier cu date de intrare pentru simulare. Pentru simplificare, valorile vertexurilor vor fi preprocesate cu ajutorul utilitarului **contour.sh** (respectiv **ImageContours.java**), care va transforma valorile imaginii în date specifice necesare vizualizării. Legătura dintre punctele obținute din determinarea contururilor va fi supusa unui proces de postprocesare în vederea eliminării punctelor aflate pe aceeași direcție. Acest lucru este deosebit de important în vederea utilizării ca deplasare liniară (G1) a sculei pe durata tăierii (respectiv gravării prin frezare a conturului). În versiunea actuală, determinarea contururilor interioare nu este implementată, principiul de funcționare fiind însa asemănător.

Pentru determinarea conturului s-a recurs la un algoritm simplificat de urmărire a contururilor. Acesta utilizează vecinătățile de tip 8, codificând direcțiile de căutare în sens orar, plecând de la valoarea 0 asignată poziției orizontale dreapta (ora 3).

Din descrierea algoritmului se poate deduce faptul că, spre deosebire de cazul aplicării transformării Hough pentru determinarea liniilor, direcția este doar cea dată de cele 8 vecinătăți - deci în pași de 45° - pe când transformarea Hough ne permite determinarea liniilor de direcții arbitrare. Datorită calculelor intensive pe de-o parte, dar și recursivității, metoda se pretează doar pentru imagini digitale de dimensiuni reduse. Un exemplu de conturare al unei astfel de zone este prezentat în exemplul următor, în care imaginea inițială este cea din Fig. 210, iar cel conținând contururile corespunzând Fig. 211.

Fig. 210: Frezare și tăiere - imaginea inițială

Fig. 211: Conturare

Rezultatul frezării, poate fi observat în imaginea alăturată.

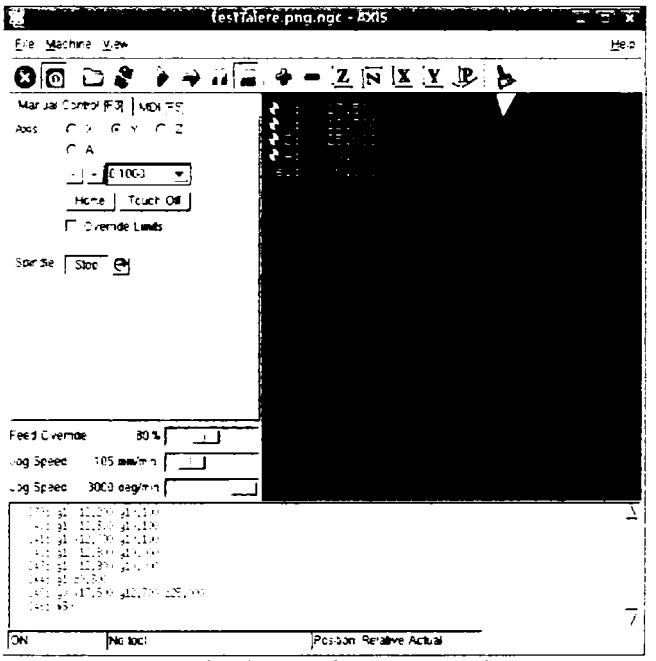


Fig. 212: Rezultatul tăierii frezării contururilor (1)

Metoda ideală – deși neimplementată în cadrul lucrării de față - este după cum se observă în acest caz o combinație între transformarea Hough pentru linii în vederea detectării acestora și detectorul Harris în vederea detectării colțurilor. Astfel pot fi eliminate segmentele de linie aflate în afara contururilor, iar urmărirea conturului se va face doar prin utilizarea acestor puncte.

Pe de altă parte, utilizarea acestei metode poate conduce la complicații suplimentare în cazul utilizării asupra unei imagini de tipul celei din figura următoare (Fig. 214), deci în care este urmărită doar tăierea sau frezarea unui anumit contur. Din acest motiv, un alt algoritm a fost implementat (având fișierul **follow.sh** ca script de start și respectiv **FollowContours.java** ca [illegible]), [illegible] limi ării ac s i problematici. Algoritmul, deși bazat pe cel anterior însă puțin modificat prin eliminarea intermediară a punctelor analizate, permite generarea de G-Code pentru astfel de situații. Aplicând algoritmul de tăiere (respectiv de urmărire a conturului), rezultatul arată ca în Fig. 213.

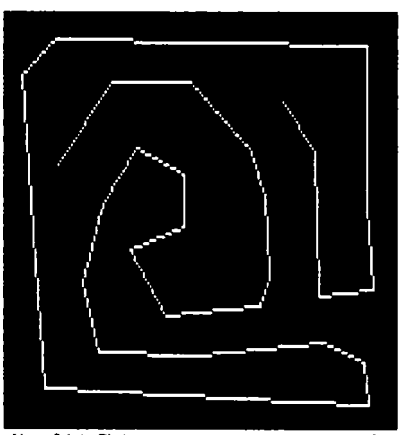


Fig. 214: Tăierea ș. urm[illegible]a contu.u.lo. sau liniilor

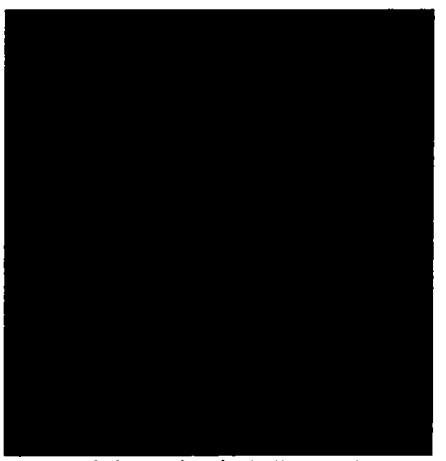


Fig. 213: Rezultatul tăierii (urmărirea contururilor liniilor)

Similar exemplului anterior, rezultatul frezării este redat în Fig. 215.

Algoritmul Haris, transformarea Hough și cel necesar generării de G-Code, deși cu siguranță mult mai performant în cazul imaginilor având numeroase linii drepte sau de dimensiuni mari, este de complexitate ri-

dicată, cel implementat în continuare fiind relativ simplu dar eficace.

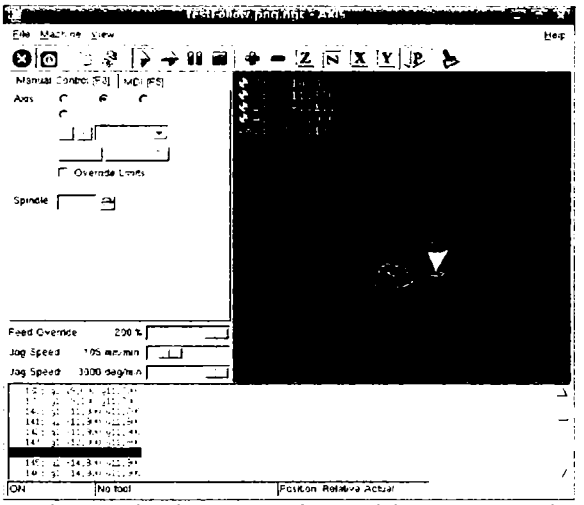


Fig. 215: Rezultatul tăierii prin algoritmul de căutare a cantelor modificat

În continuare vom utiliza ca intrare o imagine digitală în nuanțe de gri, prezentată în Fig. 216.

Fig. 216: Frezarea conturutilor - imaginea inițială

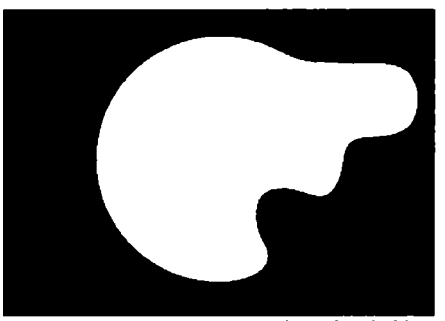


Fig. 217: Frezarea contururilor - threshold

Prin aplicarea unui nivel de threshold, și inversarea valorilor pixelilor, vom obține imaginea binară necesară în vederea prelucrării (Fig. 217). Imaginea corespunzătoare simulării frezării suprafeței astfel determinate este următoarea:

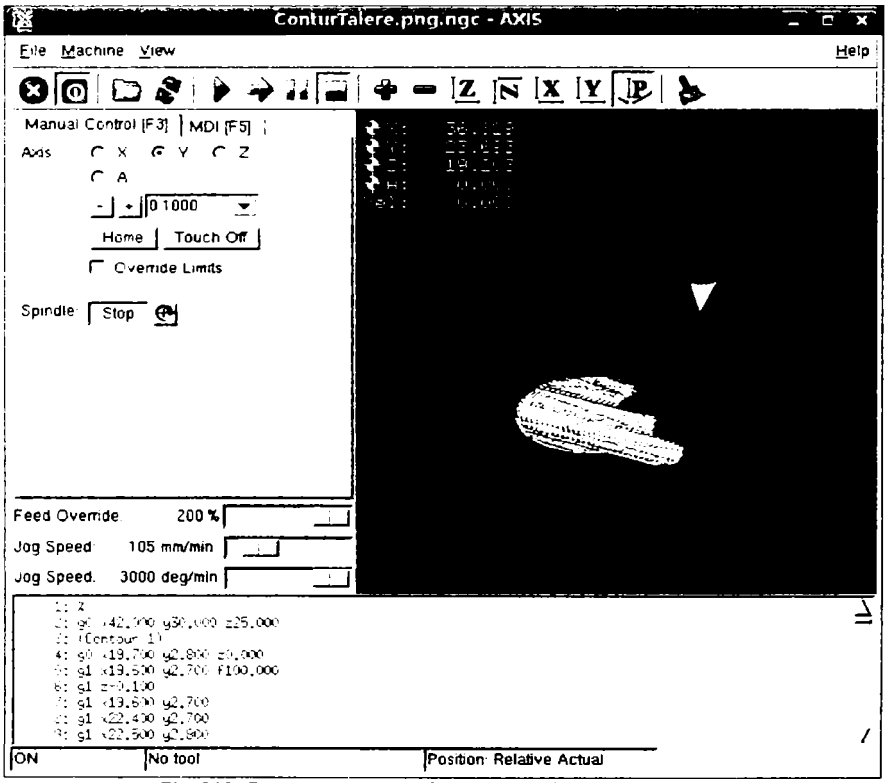


Fig. 218: Frezarea unei suprafețe rezultate din imaginea binară

Cu toate că în momentul de față traseul de deplasare al sculei este departe de a fi optimal, considerând faptul că programului nu i-au fost impuse sau sugerate moduri optimale sau preferențiale de deplasare, scopul principal al realizării suprafeței dorite este atins.

9.2.2 Frezarea unui profil rezultat din analiza unei imagini digitale

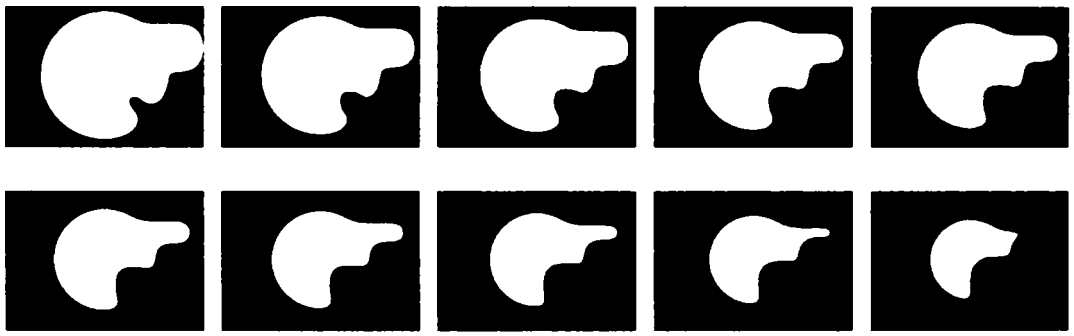
Dat fiind faptul că nu există convenții referitoare la modul de utilizare a informațiilor de foreground și background în ceea ce privește modul de prelucrare, și pe de altă parte datorită faptului că în algoritmul implementat a fost utilizată valoare 0 ca background (deci ca zonă neinteresantă pentru frezare), vom fi nevoiți în anumite condiții să inversăm valorile pixelilor astfel obținuți, în funcție de modul dorit de prelucrare a formei rezultate (pozitiv sau negativ – dependent de activitățile prevăzute ulterior). În cazul de față, vom considera că imaginea prezentată este o placă plană în care urmează să fie prelucrată o adâncitură corespunzătoare zonei întunecate.

Pentru a demonstra effectivitatea algoritmului propus, cu toate impedimentele momentan existente referitoare la modul de parcurgere al traseului sculei în vederea acoperirii suprafeței de generat, vom arăta că prin utilizarea unui script foarte simplu, un profil 2.5D poate fi generat în mod trivial. De asemenea, trebuie luat în considerare faptul că informațiile referitoare la sculă au fost ignorate, considerându-se că freza este de tip ac, deci baleierea suprafeței de frezat va avea loc pentru fiecare pixel al imaginii binare. Evident că prin modificarea programului și introducerea unei raze a frezei de lucru, numărul trecerilor va scade simțitor.

Acest script (**follow2_5.sh**) realizează prin intermediul liniei de comandă convertirea imaginii digitale într-

una binară, permițând inversarea valorilor pixelilor prin utilizarea unui parametru suplimentar care succede numele fișierului conținând imaginea inițială.

Utilizând imaginea din Fig. 216, un pas de iterare de 10% pentru nivelele de gri în vederea determinării treptei de separare și un adaos de 0.5mm pe iterație, a fost obținută secvența de imagini următoare.



Tab. 11: Secvențe threshold pentru frezarea 2.5D

Rezultatul simulării și deci a prelucrării pe freza cu comandă numerică fiind prezentat în imaginea următoare.

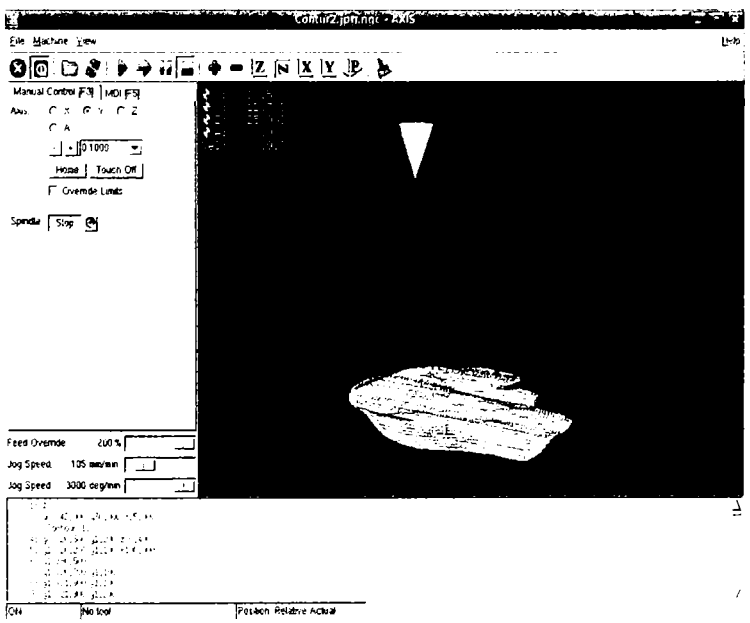


Fig. 219: Rezultatul frezării 2.5D a unei imagini în nuanțe de gri

9.2.3 Gaurire CNC multiplă a orificiilor rezultate din analiza unei imagini digitale

În vederea burghierii gaurilor de dimensiuni diferite descoperite, vom utiliza ca exemplu atât rezultatul a transformării Hough pentru cercuri cât și a unui algoritm bazat pe pattern matching, datorită faptului că imaginile de analizat sunt binare.

Pentru ambele metode, au fost utilizate câte 1100 de imagini digitale de dimensiune de 300x200, generate aleatoriu cu cercuri de raze între 20 și 30 de pixeli, rezultatele determinării centrelor cercurilor fiind pre-

zentate în tabelul următor:

	Hough pentru cercuri	*Pattern Matching*	*Metoda modificată*
Timp de rulare	0:41:54.629	1:32:11.623	0:48:0.188
Cercuri determinate din 3300	2883	3300	3265
Abatere procentuală	12.64%	0%	1.06%

Tab. 12: Comparație Hough pentru cercuri / Pattern Matching /Metoda Hough modificată

Dat fiind faptul că poziția centrelor în cadrul imaginii a fost determinată în mod aleatoriu prin program, suprapunerea cercurilor astfel generate n-a mai fost eliminată, deci apariția unor eventuale abateri de la 100% este explicabilă și prin acest fapt. Pe de altă parte, un alt rol foarte important îl joacă faptul că, datorită condițiilor date, prin suprapunerea parțială a cercurilor, maximele spațiului Hough sunt minimal perturbate. Această perturbație este însă suficientă pentru a caracteriza un singur punct ca centru, deoarece acumulatorului i se vor adăuga puncte suplimentare apărute numai din cauza suprapunerii cercurilor căutate. Dacă în schimb, în locul utilizării valorii de prag maxime se va recurge la alegerea unei trepte mai mici, riscul apariției centrelor false sau duplicate este mult prea mare pentru a nu se recurge la metode suplimentare de filtrare a rezultatului – aceste metode ducând de asemenea la creșterea timpului de execuție a programului.

Metoda este totuși imbatabilă în cazul în care cercurile căutate sunt cu siguranță separate, valoarea de maxim nemaifiind influențată de valori suplimentare datorate cercurilor interioare sau întrepătrunse.

În vederea menținerii generalității și deci considerând că imaginile digitale nu vor fi în general perfecte, în algoritmul propus în continuare, poziția centrului n-a fost restricționată printr-o egalitate absolută, ci valoarea determinata a fost considerată ca fiind cea căutată în condițiile în care centrul determinat este în interiorul vecinătății 8, deci într-unul dintre pixelii învecinați centrului real. Din acest motiv, valoarea determinată a centrelor este uneori în vecinătate de un pixel. Gradul de eroare nu poate din nefericire fi diminuat, dat fiind faptul că pixelii unei imagini digitale sunt valori discrete întregi, iar operațiile cu numere întregi induc inevitabil erori de aproximare în cazul aplicării operațiilor statistice asupra valorilor existente. Cu toate acestea însă, considerând valoarea unui pixel ca fiind corespunzătoare unui micron sau a unei sutimi de milimetru, abaterea astfel rezultată este acceptabilă.

În cazul algoritmului de pattern matching, este evident că toate cercurile au fost determinate, dar în detrimentul timpului necesar. De asemenea, o implementare a algoritmilor prezentați într-un limbaj neinterpretativ (e.g. C sau C++) și optimizarea codului va aduce cu sine o creștere drastică a timpului necesar determinării centrelor.

9.2.3.1 Utilizarea transformării Hough generalizate modificată pentru cercuri

În vederea detectării cercurilor a fost utilizat algoritmul prezentat în anexă, dar într-un limbaj formal, el poate fi exprimat după cum urmează:

```
pentru fiecare punct semificativ al imaginii de analizat
  pentru fiecare punct semificativ al referinţei
    se incrementează valoarea acumulatorului relativ la punctul imaginii
se aplică treapta dorită asupra acumulatorului
pînă cînd acumulatorul mai conţine puncte semificative
  copiază acumulatorul
  se numără obiectele din acumulator
  se aplică operaţia de erodare asupra copiei
  se numără obiectele din copia erodată
  se compara numarul obiectelor
  dacă numărul este diferit
```

```
    se calculează poziţia centrului ca medie aritmetică a coordonatelor
    se salvează poziţia centrului în lista de centre determinate
  copia devine acumulatorul actual
pentru fiecare punct din lista de centre
  verifică valabilitatea centrului prin compararea cu imaginea iniţială
  dacă centrul nu este valabil
    elimină centrul actual din lista centrelor
```

Deşi algoritmul prezentat este bazat pe transformarea Hough generalizată, după cum se poate observa, utilizează nu numai acumulatorul specific transformării amintite, ci printr-o variaţie a algoritmului de ultima eroziune permite determinarea poziţiei centrelor indiferent de nivelul treptei alese. Faptul că în vederea determinării poziţiei centrului media aritmetică este utilizată asupra coordonatelor discrete ale acestuia, erorile de rotunjire sunt inevitabile, aducând cu sine o deviere a acestuia. În vederea eliminării acestui impediment şi în cazul în care precizia determinării centrului este primordială, este recomandabilă adaptarea calculului poziţiei centrului prin metode de căutare a maximumului local în acumulatorul iniţial (înaintea aplicării treptei) şi/sau prin adaptarea algoritmului de verificare a validităţii centrelor determinate – ambele însă conducând la operaţii suplimentare de calcul şi deci la creşterea timpului de rulare.

Fig. 220: Imagine binara conţinând contururi de orificii de raze diferite

Prin utilizarea algoritmului aminitit auspra imaginii anterioare, spaţiul Hough determinat va avea reprezentarea următoare. Dupa cum se observă, datorită contururilor clare şi bine definite, spaţiul astfel calculat (în urma aplicării operaţiei de treaptă - threshold) este constituit din puncte clar determinate, reprezentând centrele cercurilor căutate.

Fig. 221: Spaţiul Hough normalizat aferent imaginii anterioare

Programul implementat generează de asemenea imaginea finală, compusă din cea iniţială peste care au fost trasate cu linie roşie cercurile detectate - a căror poziţie a fost determinată prin utilizarea algoritmului amintit (Fig. 222).

Această metodă de lucru este foarte utilă în cazul testării algoritmului implementat, datorită faptului că rezultatele finale pot fi puse în evidență atât prin afișarea lor sub forma de coordonate, precum și sub formă

Fig. 222: Rezultatul determinării centrelor cercurilor

de imagine digitală atât a spațiului Hugh determinat cât și a suprapunerii rezultatelor peste imaginea inițială în vederea comparării vizuale. Prima versiune, cea a afișării coordonatelor, ne permite pe de altă parte atât implementarea de subprograme de test automatizat, cât și primul pas în direcția paralelizării căutării și determinării cercurilor.

Exemplul următor este bazat pe analiza unei imagini reale, prelucrate digital în vederea determinării pozițiilor centrelor cercurilor:

Prin aplicarea unuia din filtrele de determinare a contururilor amintite, inclusiv transformarea imaginii

Fig. 223: Imaginea originală

Fig. 224: Imaginea transformată binar și cu contururi deteminate

intr-una conținând doar nuanțe de gri, imaginea originală din Fig. 223 va deveni imaginea de intrare în vederea detectării cercurilor (Fig. 224).

Prin aplicarea succesivă a algoritmului anterior, pentru cercuri cu raze variabile, vom obține pas cu pas rezultatele intermediare prezentate în imaginile următoare. Evident, că imaginile pentru care n-au fost detectate cercuri in imaginea anterioară, nu au fost luate în considerare și deci nici nu vor fi afișate.

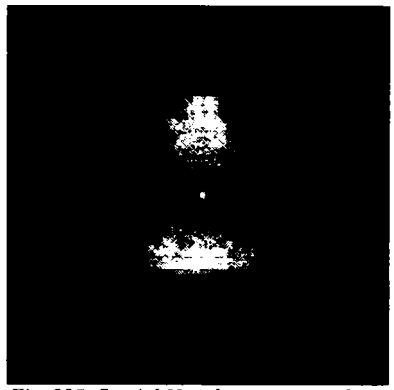


Fig. 225: Spaţiul Hough pentru raza de 11

Fig. 226: Cercuri cu raza de 11 determinate

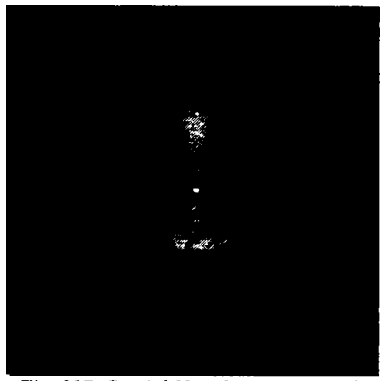


Fig. 227: Spaţiul Hough pentru raza de 18

Fig. 228: Cercuri cu raza de 18 determinate

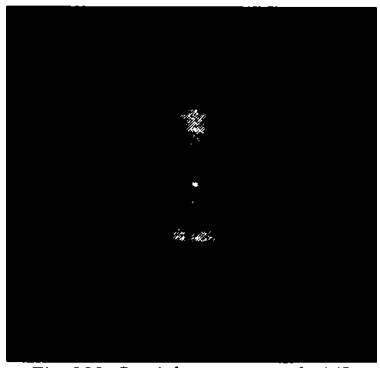


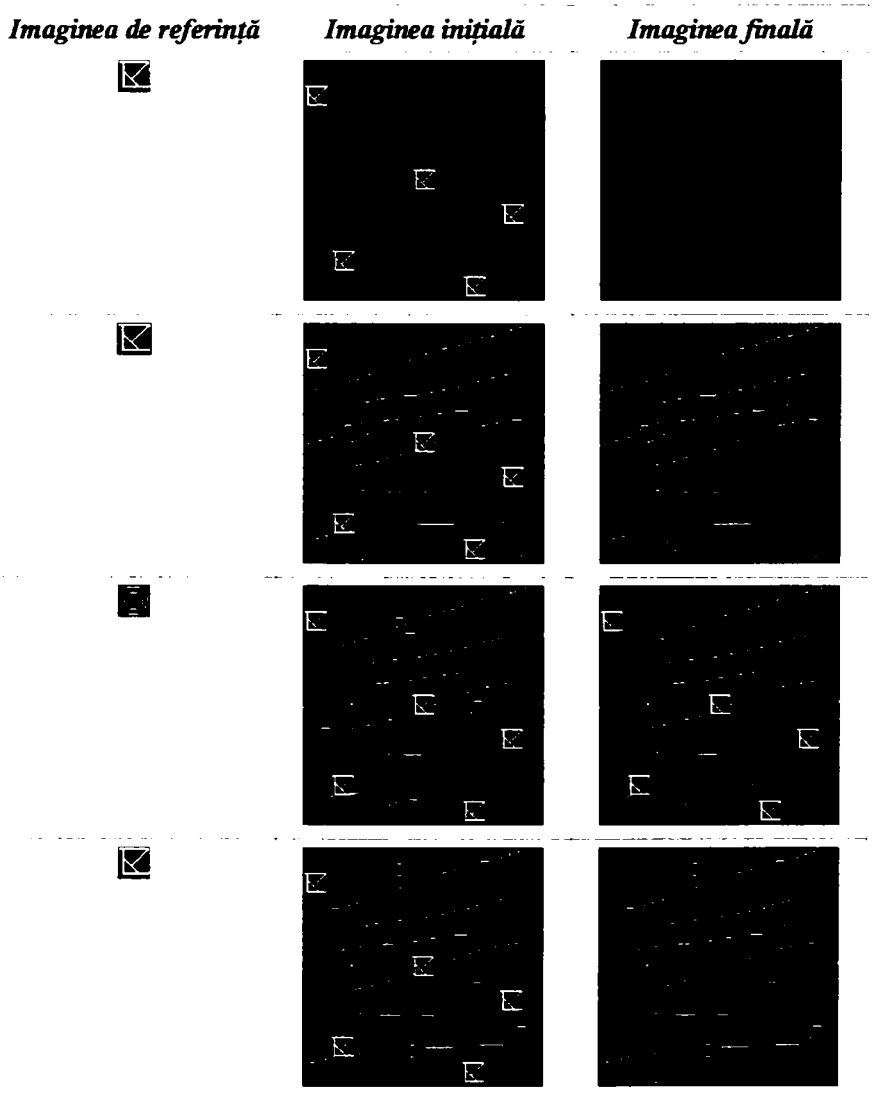
Fig. 229: Spaţiul pentru raza de 147

Fig. 230: Cercuri cu raza de 147 determinate

Suprapunerea cercurilor astfel determinate peste imaginea iniţială este de asemenea implementată, va fi însă prezentată în cadrul exemplului de distribuire în GRID a algoritmului, caz în care această suprapunere are o importanţă deosebită, fiind absolut necesară în vederea obţinerii rezultatului final.

9.2.3.2 Utilizarea algoritmului Hough generalizat pentru alte forme

Datorită faptului că algoritmul nu este specific cercurilor, ci utilizează doar imagini de referință din imagini binare, este fără modificări utilizabil pentru detecția oricăror tipuri de contururi. Scopul implementării și prezentării pentru cercuri a fost în principal al comparării metodelor consacrate cu cea prezentată. Din acest motiv, și pentru simplificare, doar câteva imagini exemplare vor fi afișate în continuare, lista posibilităților fiind însă nelimitată.



Tab. 13: Metoda General Hough modificată – exemple

9.2.3.3 Simularea găuririi unei plăci în urma analizei unei imagini binare

În vederea simulării gauririi unei plăci în urma analizei imaginii digitale, vom recurge la generarea aleatorie a unei imagini binare cu cercuri de raze diferite, dintre care doar anumite raze vor fi considerate ca interesante pentru gaurire. Imaginea binară generată, şi rezultatul căutării centrelor centrelor prin aplicarea algoritmului Hough generalizat, este prezentată în Fig. 231 iar rezultatul simulării prelucrării în figura următoare.

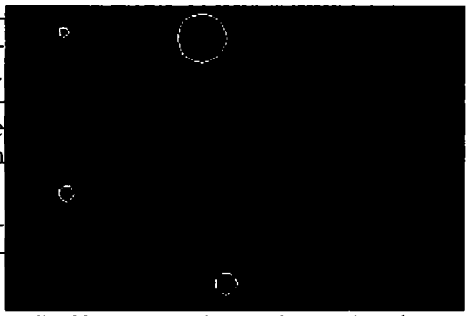


Fig. 231: Imaginea binară aleatorie în vederea simulării gauririi

În urma rulării programului de detectare a centrelor, un fişier cu extensia „.radii" a fost generat, având conţinutul următor:

```
9- -- --
237 54 20
206 137 20
```

După cum putem observa, algoritmul nu ia în considerare modul de reprezentare al coordonatelor în cele două sisteme, cel al imaginii digitale şi cel oferit de Blender. În cazul imaginilor digitale, originea axelor de coordonate este situată în colţul din stânga sus iar valorile lui y cresc pe măsură ce coborâm în jos de-a lungul axei, pe când în sistemele de CAD (şi deci şi în cazul lui Blender) originea coordonatelor este situată în colţul din stânga jos, valorile lui y crescând în sus. Din acest motiv, pentru o vizualizare corectă, o oglindire a coordonatelor este necesară, fie printr-o rotaţie de 180° în jurul axei x şi o translatare pe y cu o măsură egală cu înălţimea imaginii considerate, fie prin rotaţia în jurul unei paralele la axa x prin mijlocul înălţimii imaginii.

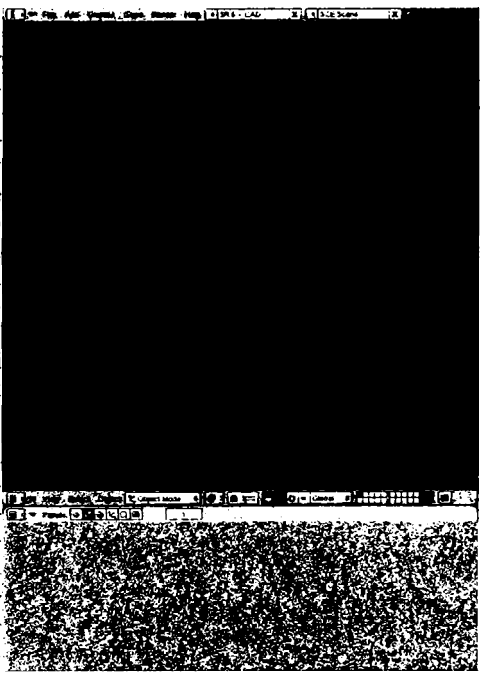


Fig. 232: Rezultatul simulării gauririi în Blender

Programul implementat şi prezentat în anexă, primeşte ca parametri de intrare doar coordonatele centrelor găurilor de prezentat şi diametrele acestora, fapt pentru care adaptarea acestor coordonate nu afectează generalitatea modelului – cu atât mai mult cu cât nici dimensiunile plăcii utilizate nu sunt luate în considerare, în scopul acestei lucrări fiind considerate ca având o valoare fixă, dată printr-un membru al clasei de generare a simulării. În cazul în care programul determinarea centrelor sau un modul de postprocesare realizează transformarea de coordonate, evident că şi reprezentarea prelucrării va deveni acurată.

În vederea simulării, reprezentarea poziţiei şi mărimii orificiilor este suficientă, fapt pentru care o complicare a programului implementat în vederea realizării operaţiilor booleene asupra corpurilor implicate n-a mai fost luate în considerare. Generarea programului G-Code aferent este însă importantă, fapt pentru care o vom expune în exemplul următor.

```
%
g0 x0.000 y0.000 z20.000
(circle 1)
g0 x9.300 y6.100 z2.000
g1 x9.300 y6.100 z1.000 f10.000
```

```
g1 x9.300 y6.100 z-3.000
g0 x9.300 y6.100 z2.000
(circle 2)
g0 x23.700 y5.400 z2.000
g1 x23.700 y5.400 z1.000 f10.000
g1 x23.700 y5.400 z-3.000
g0 x23.700 y5.400 z2.000
(circle 3)
g0 x20.600 y13.700 z2.000
g1 x20.600 y13.700 z1.000 f10.000
g1 x20.600 y13.700 z-3.000
g0 x20.600 y13.700 z2.000
g0 x0.000 y0.000 z20.000
m30
```

Scriptul python aferent, necesar simulării și generării programului G-Code este prezentat în anexă, o descriere amănunțită nefiind necesară având în vedere simplitatea sa - determinată în special de faptul că grosimea plăcii a fost considerată ca fiind de 1mm, burgierea este efectuată cu un burghiu de 2mm (deci totul la o scară imagine/BlenderUnits de 10/1).

9.2.4 Exemplul simulării frezării unui model de ventil cilindric

După cum am mai amintit, problema principală a utilizării programului Blender ca interfață CAD, este faptul ca aceste utilizează așa-numitele 'Blender Units' în vederea definirii obiectelor. În vederea simplificării algoritmilor, în această versiune toate obiectele vor fi modelate fără rotație sau translație (așadar cursorul 3D va fi poziționat înainte de generarea obiectelor). Astfel, coordonatele obiectelor NURBS pot fi citite direct utilizând tabela de proprietăți a obiectului activ ('Transform Properties'), dat fiind faptul că problema ce urmează a fi rezolvată este cea de a genera programul G-Code necesar urmăririi cu scula de prelucrat a curbelor modelate.

Din ace oti , - sidera scula de prelucrare ca fiind o freza sferică. Versiunea actuală a programului presupune de asemenea că freza este reprezentată de un obiect numit 'Mill' iar curba care va determina traiectoria de prelucrare este selectată interactiv.

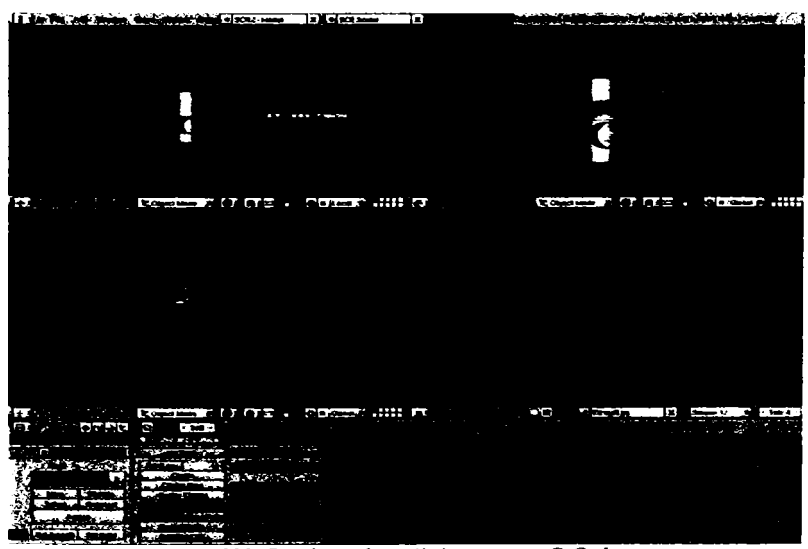


Fig. 233: Simularea frezării și generarea G-Code

Transformarea unităților de lucru între 'Blender Units' și coordonatele mașinii (milimetri) este efectuată static în interiorul scriptului python printr-o simpla scalare. În urma simplei porniri a scrip ului (<ALT>+P) avem posibilitatea simulării mișcării frezei, fie sub formă de animație, fie ca secvență pas cu pas. Dat fiind faptul că versiunea actuală nu permite încă utilizarea celei de-a patra axe (de rotație) programul poate fi utilizat doar pentru urmărirea curbelor generice pentru mașini cu trei axe.

Datele rezultate însă, adaptate (manual) pentru cea de-a patra axă, au condus la realizarea ventilului din imaginea redată în Fig. 234 (Plexiglas pe o maşina de tip UNI-Mill).

Fig. 234: Rezultatul frezării în plexiglas

9.2.5 Simularea frezării şi generarea de G-Code din curbe generalizate

O altă proprietate foarte importantă oferită de Blender este faptul că curbele dorite pot fi generate prin program şi nu doar interactiv. Astfel, în cazul în care este necesară frezarea unei suprafeţe matematice, sau determinate experimental, datele acestei suprafeţe pot fi introduse prin intermediul unei interfeţe python în Blender. Curbele astfel generate pot fi cu uşurinţă selectate în buclă tot prin intermediul programului, iar generarea G-Code este apelată pentru fiecare dintre curbele astfel selectate, obţinându-se un program CNC com- let, rulabil e ma ini cu comanda numerică. Ca exemplificare, programul anterior a fost modificat în vederea simulării unei prelucrări freză dege. Sing r de ebire e e t faptul că freza este reprezentată printr-un cilindru (tot denumit "Mill") iar originea acesteia nu este centrul obiectului (ca în cazul sferei) ci centrul bazei inferioare. Selectând curba dorită şi pornind scriptul, suprafaţa grafică va fi asemănătoare celei din Fig. 235. Deşi simplă, interfaţa grafică oferită este suficientă şi complet integrată în Blender, iar după cum am demonstrat, ea permite atingerea scopului dorit.

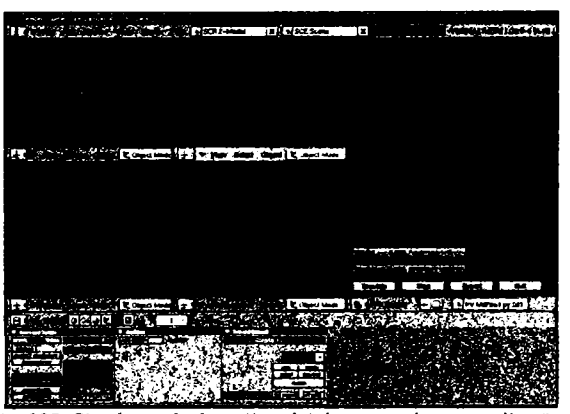


Fig. 235: Simularea deplasării sculei după o curbă generalizată

Modulul principal Blender.Draw utilizat în designul interfeţei HMI este prezentat în tabelul următor, şi reprezintă totalitatea elementelor necesare şi posibile de folosit în vederea generării interfetelor utilizator:

Button(text, event, x, y, w, h)	Buton care apăsat lansează o comandă în momentul eliberării. "text" este eticheta (textul) butonului, "event" reprezintă numărul evenimentului produs, "x" şi "y" sunt evident coordonatele de start ale butonului iar "w" şi "h" corespund dimensiunii acestuia
Toggle(text, event, x, y, w, h)	Un buton de tip toggle care-şi pastrează proprietatea de a fi apăsat sau nu. Parametri necesari sunt identici cu cei din cazul tipului Button
val=Slider(text, event, x, y, w, h, val, min, max)	Control de tip cursor. "val" reprezintă valoarea curentă, "min" şi "max" limitele de valori ale controlului. Acest element de control incorporează şi un element de tip Number prin care valoarea curentă poate fi introdusă direct. Această valoare este returnată în variabila "val"
n=Number(text, event, x, y, w, h, val, min, max)	Control pentru introducerea numerelor. Parametrii sunt identici cu cei din cadrul controlului Slider
s=String(text, event, x, y, w, h,val)	Control pentru introducerea de valori de tip text. "val" reprezintă textul iniţial, iar parametri sunt asemenea celor din cazul controlului de tip Button
Menu(text, event, x, y, w, h, val)	Control de tip Meniu. Opţiunile sunt introduse ca text în variabila "text" în formatul următor: [Titlu%t|Optiune1%x1|Optiune2%x2|...Optiune N %xn]
Redraw()	Metoda ce poate fi apelată în vederea actualizării ferestrelor Blender în următorul moment posibil
Text(text)	Afişează textul dorit în GUI
Exit()	Ieşire (terminare) GUI şi redarea controlului în Blender
Register(gui, event , button)	Prin această metodă o funcţie der tip "gui" este registrată în vederea actualizării. "Event" reprezintă funcţia ce va fi apelată în vederea manipulării de evenimente (e.g. apăsări de taste), iar "button" pentru cele specifice elementelor de control prezentate anterior.

Tab. 14: Modulele de baza ale pachetului BlenderDraw

În vederea utilizării elementelor prezentate anterior, şi deci a programului implementat, este necesar ca interpretorul python să fie instalat pe sistem, iar modulele Blender specifice utilizării interfeţei python să fie de asemenea instalate, o foarte bună documentaţie fiind prezentată pe site-ul utilitarului.

9.3 Simularea strunjirii

Deşi simularea frezării este un domeniu foarte important, iar urmărirea curbelor generalizate rezultate din modelare şi cu generarea programelor NC necesare prelucrărilor pe maşini cu comandă numerică rezultate nu poate fi contestată ca importanţă, în cele ce urmeaza vom încerca să demonstrăm că şi strunjirea poate beneficia din plin de facilităţile oferite de Blender.

9.3.1 Consideraţii generale

În vederea simplificării structurii programului, vom considera din nou faptul că un milimetru real corespunde unei unităţi Blender, fără a reduce prin aceasta gradul de generalitate a consideraţiilor următoare.

Modulul de simulare a fost conceput ţinând cont de premisa principală a lucrării de faţă, şi anume de faptul că softul trebuie să fie cât mai simplu pentru a fi aplicat şi testat exemplar pe majoritatea maşinilor existente. Pe de altă parte, strungul cu comandă numerică utilizat în vederea validării simulării este un 'UNI-Turn 2000' produs de firmele Sherline [58] şi TheCoolTool[59]. Sistemele de operare testate în acest scop au fost Puppy-Linux, BDI 4.3 şi Ubuntu 6.06 LTS, toate utilizând EMC2.0[60].

Ţinând cont de impedimentele maşinii amintite, am considerat că punctul de zero al maşinii coincide cu originea axelor din Blender, programul calculând doar mişcările de-a lungul curbelor generate. Această presupunere simplifică semificativ poziţionarea camerei, permiţându-ne salvarea configuraţie preferate ca default (CTRL+U) pentru toate simulările.

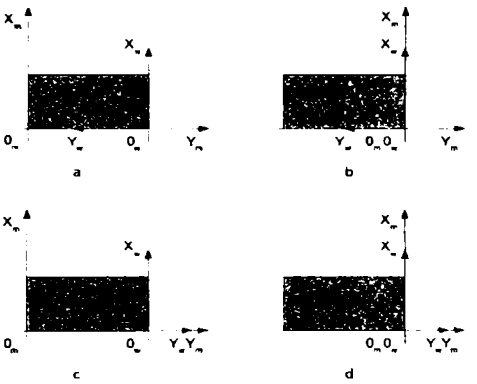


Fig. 236: Sisteme de coordonate

Din punct de vedere matematic, este cu siguranţă mult mai uşoară implementarea prin deplasarea relativă a originii pe maşina de lucru, dar dat fiind faptul că strungul utilizat rămâne în general acelaşi şi împreună cu el şi dimensiunile maxime ale pieselor de prelucrat, de cele mai multe ori obiectul rezultat va ieşi din zona focală a camerei, aceasta trebuind recalibrată pentru fiecare piesă simulată în parte. Din acest motiv am hotărît implementarea simulării în funcţie de condiţiile reale existente, şi deci deplasând originea piesei – această deplasare fiind reprezentată printr-un plan. Figura alăturată ilustrează diferenţa dintre cele două situaţii prezentate anterior. Poziţiile b şi d sunt cu siguranţă mai uşor de implementat, însă utilizând versiunea a, toate celelal·e po· fi uşor calcula·e prin in·ermediul unui offse· şi a orientării axelor corespunzătoare reprezentată prin valori de 1 sau -1.

Cu toate că strunjirea necesită doar deplasări 2D, modulele implicate au fost concepute în 3D cu scopul unei adaptări mai rapide pentru simularea ulterioară a strunjirii. Din acest motiv, punctele de offset şi orientările axelor sunt reprezentate prin vectori 3D. Pentru moment, axele suplimentare posibile (A, B, C, R, U, V) nu au fost implementate. Totodată, ţinând cont de faptul că lucrarea de faţă nu urmăreşte implementarea completă a conceptelor urmărite ·i î· ·-i·--l ·â·d ·--tă ·ă d·-·-·-t·z· ·i·bilit·· ·--·-ceptelor propuse (proof of concept), geometria sculei nu este luată în considerare. Geometria necesară este trivială şi cunoscută, iar rutinele de intersecţie a poligoanelor nu sunt restricţionate în ceea ce priveşte numărul de vertice implicate, deci generalitatea conceptului nu este prin eliminarea acestor considerente restrânsă. Pentru exemplificare însă, în figura alăturată este prezentată schematic diferenţa dintre soluţia considerând doar punctele semificative şi cea cu implicarea geometriei sculei aşchietoare.

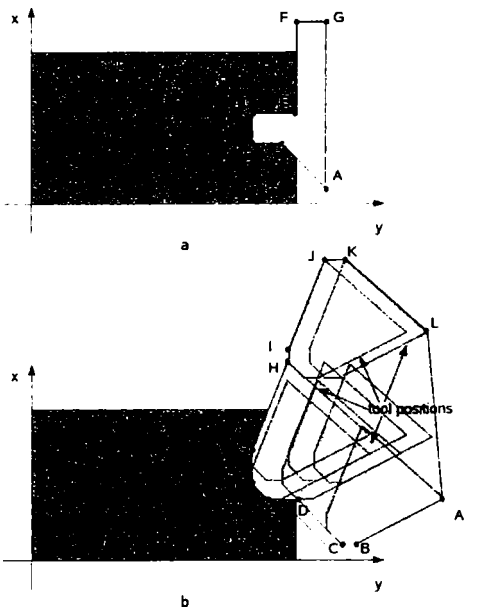


Fig. 237: Aplicarea geometriei sculei

După cum se poate observa, diferenţa principală constă în calcularea geometriei noului poligon considerând geometria sculei implicate şi a punctului de referinţă al acesteia. În cazul exemplului de mai sus, punctul de referinţă al sculei este considerat ca fiind situat la intersecţia tangentelor la sculă, tangente paralele cu axele de coordonate x şi y. Evident că poligonul rezultat este foarte diferit de cel anterior, care considera doar deplasările pure din G-Code, însă dat fiind faptul că intersecţia poligoanelor nu depinde de numărul punctelor de contur, generalitatea nu este afectată.

Pentru moment, doar deplasările G-Code şi piesele intermediare vor fi reprezentate pe parcursul simulării în Blender, însă dezvoltarea ulterioară a programelor va permite generarea de curbe interpolate (IPOs) în Blender în vederea obţinerii de animaţii ale procesului de aşchiere – în special prin interpretarea comenzii

F din fişierul NC de intrare. De asemenea, datorită modulelor deja existente de mecanica fluidelor, implementarea comenzilor de control a răcirii (M7/M8/M9) şi reprezentarea lor în cadrul simulării va fi de asemenea posibilă.

Tot datorită premiselor prezentate anterior, în versiunea de faţă doar interpolările liniare au fost luate în considerare, cele circulare fiind însă prevăzute, iar implementarea lor fiind realizată într-un viitor foarte apropiat.

Singura dificultate de implementare va fi cu siguranţă cea corespunzătoare comenzii G33 (spindle synchronized motion) care necesită programarea de noi rutine de generare în vederea simulării. Dat fiind faptul că în momentul de faţă viteza de aşchiere este ignorată, iar maşinile cu comandă numerică utilizate nu au posibilitatea varierii vitezei de rotaţie a axei principale prin intermediul comenzilor numerice, aceste comenzi au fost de asemenea ignorate. Toate celelelte comenzi din specificaţie codului NIST RS274NGC [61], cum ar fi modul de adresare al distanţelor (absolut 90 sau relativ 91) precum şi schimbarea unităţilor de lucru (inch sau milimetrii) nu prezintă complicaţii de implementare.

9.3.2 Modulele Python implementate

LatheSim – Acest modul conţine rutinele de bază ale simulării strunjirii. Manipulează intern datele comune sub formă de vertice şi feţe, stocându-le în acelaşi timp şi pe cele referitoare la traiectoriile de lucru şi deplasările rapide. Acest modul este constituit din următoarele clase:

- **LatheSimMaths** – clasă ce se ocupă de operaţiile matematice şi geometrice necesare stocând obiectele obţinute în urma procesului de parsing, respectiv de interpretarea codului NC de intrare

- **Utils** – o clasă conţinând utilitarele necesare manipulării de vertice şi feţe

NCTools – acest modul reprezintă parserul NC al fişierului ".nc" de intrare, intrepretând comenzile citite în obiecte interne uşor utilizabile ulterior de către interfaţa Blender. Clasele conţinute în acest modul sunt:

- **Path** – reprezentarea internă a unui traseu NC determinat printr-o secvenţă G-Code

- **NCParser** – parserul care analizează şi transformă fişierul NC de intrare. Scopul acestei clase este de a interpreta comenzile G-Code în vederea transformării lor în trasee poligonale necesare intersecţiei cu semifabricatul ori piesa intermediară, sau pur şi simplu pentru generarea traseului sculei. Această clasă poate de asemenea fi utilizată în scopul generării de curbe interpolate (IPO) în Blender, în vederea animării succesiunilor de operaţii

NCBlender – implementează interfaţa Blender cu utilizatorul, combinând datele calculate de parser din fisierul ".nc" de intrare cu modulele API din Blender

BlenderNCLatheSim – Simularea procesului de prelucrare poate fi realizată fie prin încărcarea fişierului Python "**BlenderNCLatheSim.py**" - după trecerea în modul corespunzător de reprezentare (scripts window), fie prin încărcarea şi rularea automată prin pornirea aplicaţiei utlizând numele fişierului ca şi parametru de intrare, ca în exemplul liniei de comandă de mai jos:

```
blender -P BlenderNCLatheSim.py
```

Datorită modului special de construcţie a acestui script, utilizatorul poate porni deci aplicaţia de simulare chiar şi din interiorul IDE-ului preferat în vederea dezvoltării de programe (e.g. Eclipse, IDLE, NEdit, SPE sau chiar şi vi). Unul dintre cele mai interesante şi practice editoare este SPE, în special în vederea testării programelor scrise pentru Blender, datorită faptului că acesta poate fi pornit din interiorul lui Blender utilizând meniul **Scripts->System->Interactive Console**.

După startarea scriptului sau a programului, fişierul 'NCSample.nc' situat în directoarea activă va fi încărcat şi simulat. Conţinutul G-Code al acestuia a fost special conceput pentru a respecta necesităţile acestei lucrări, fiind foarte simplu, uşor de înţeles şi deci având un caracter didactic. Datorită acestui fapt, modul

de funcţionare al programului este intuitiv, fără a necesita cunoştinţe profunde nici în NC/CNC, nici în limbajul de programare Python şi nici în modul în care Blender utilizează acest script în vederea realizării simulării strunjirii:

```
 1: %
 2: (NC Sample)
 3: (#1 5 # - workpiece length)
 4: (#d 4 # - workpiece diameter)
 5: g0 x3 y6
 6: g1 x1.8 y5 f50
 7: g1 y0
 8: g1 x3
 9:
10: g0 y6
11: g1 x1.5
12: g1 y1
13: g1 x1.9
14:
15: g0 y6
16: g1 x1.2
17: g1 y2
18: g1 x1.5
19:
20: g0 y6
21: g1 x1.0
22: g1 y3
23: g1 x1.2
24:
25: g0 y6
26: g1 x0.8
27: g1 y4
28: g1 x1.0
29:
30: g0 y6
31: g1 x0
32: g1 x1 y4.5
33:
34: g0 y6
35: m30
```

Unul dintre specificele acestui mic program este de observat în comentariile din liniile 3 şi 4. Marca de comentar "(#" (aşa numitul "tag") este utilizată pentru a transmite parserului informaţii referitoare la lungimea semifabricatului şi diametrul iniţial al acestuia, informaţii care nu pot fi calculate în urma analizei fişierului de intrare, dar care sunt necesare simulării. În momentul de faţa doar dimensiunile iniţiale ale semifabricatului sunt utilizate, dar acest tag poate fi extins în vederea transferului de informaţii referitoare la unitatea de măsură utilizată, viteza de deplasare pentru G0, culorile şi materialele utilizate, gradul de transparenţă sau imaginea din background.

9.3.3 Descrierea interfeței grafice

În vederea simulării, Blender a fost astfel configurat încât după pornire, interfața sa (GUI) să fie constituită doar din vederea camerei (inclusiv poziția și setările corespunzătoare) și cea a elementelor de control, setare prezentată în Fig. 238.

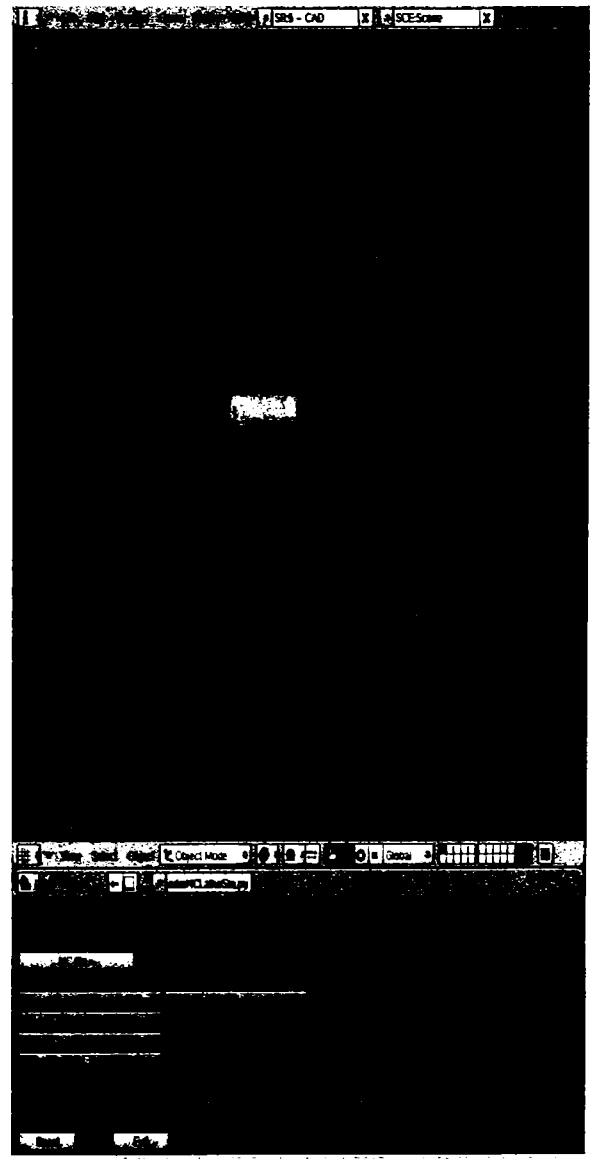


Fig. 238: Imaginea de start a simulării strunjirii

Semifabricatul și planul originii mașinii sunt de asemenea prin convenție definite ca fiind vizibile, aceste vizibilități putând fi schimbate interactiv în timpul rulării simulării. Ferestrele prezentate sunt de asemenea doar ca preferință astfel definite, forma, numărul, poziția și orientarea lor fiind de asemenea modificabilă în cursul rulării.

Figura următoare exemplifică simularea aceluiași fișier de intrare după separarea vederii principale în două și configurarea acestora într-o vedere ortogonală dreaptă și o vedere în perspectivă. Această funcționalitate, fiind integrată în Blender, poate fi utilizată oricând și în orice scop, indepentent de simularea efectuată, permițându-ne astfel scalări, rotații și modificări ale parametrilor statici ai simulării în orice combinație dorită (extrudarea pieselor intermediare, modificarea materialelor, a transparenței, a modului de shading și iluminare, a poziției camerei sau chiar animarea rezultatului). Toate aceste aspecte sunt însă Blender-specifice, o bună cunoaștere a acestuia fiind premisa principală în vederea obținerii rezultatelor dorite.

În mijlocul ecranului, semifabricatul este reprezentat ca un cilindru în albastru deschis, iar planul de offset de-a lungul axei y este reprezentat printr-un obiect Blender de tip plan, având culoarea verde.

Materialele utilizate prezintă transparență, în vederea permiterii analizării prin suprapunere a tuturor trecerilor intermediare. În momentul de față, este posibilă fie suprapunerea tuturor trecerilor, fie reprezentarea unei singure treceri. Modificarea suprafeței GUI și a programului prin includerea unui parametru suplimetar, referitor la numărul de treceri ce urmează a fi suprapuse, este însă trivială. Deoarece atât semifabricatul inițial cât și planul de offset și piesele rezultate din trecerile intermediare pot fi ascunse în decursul simulării, este posibilă atât simularea pas cu pas a prelucrării, cât și vizualizarea individuală a fiecărei secvențe de simulare.

Prin simpla apăsare a testei **F12**, simularea va fi interpretată grafic (rendered) într-o imagine conținând semifabricatul inițial, piesele rezultate din trecerile intermediare și cea finală, planul de offset și benzile corespunzătoare deplasărilor rapide și de lucru utilizate. În cazul exemplului prezentat, imaginea arată ca în figura alăturată:

Vizualizarea completă atât a traiectoriilor sculei cât și a pieselor intermediare are însă importanță doar în cazul în care numărul obiectelor intermediare generate este redus. De menționat este însă faptul că, per default, aceasta este configurația actuală a programului. După generarea tuturor obiectelor, putem însă utiliza GUI în vederea configurării modului dorit de vizualizare a rezultatelor sau încărcarea unui nou fișier.

Prin apăsarea butonului **NC-File**, binecunoscutul dialog de deschidere al fișierelor va fi startat, permițându-ne deschiderea unui nou fișier der tip ".nc", al cărui conținut va fi încărcat, analizat și simulat.

Următoarea pereche de 'Sliders' corespunde dimensiunilor inițiale ale semifabricatului și permite modificarea acestora în mod interactiv. Dacă vom seta aceste valuri cu x=1 și y=5.5, în urma apăsării tastei **F12** vom obține rezultatul simulării prezentat în figura următoare.

După cum putem observa, majoritatea traseelor sculei vor fi situate în afara semifabricatului, din cauza faptului că informația introdusă interactiv este tratată cu un grad de preferință mai ridicat decât cea existentă în fișierul de intrare (cea din urmă fiind efectiv ignorată, valorile din fișierul de intrare fiind recitite doar în momentul redeschiderii fișierului '.nc' corespunzător). Setând valoarea 1 pentru 'sliderul' offsetului semifabricatului – situat sub cel corespunzător dimensiunii – și în urma apăsării din nou a tastei **F12**, rezultatul procesului de render este redat în Fig. 242.

În vederea obținerii unei calități mai ridicate a suprafețelor obiectelor generate prin simulare, este posibilă modificarea numărului de fațete prin care o suprafață cilindrică este reprezentată în Blender. Valoarea minimă setabilă este de trei, iar cea maximă de 96, dat fiind faptul că prin creșterea numărului de fațete peste această limită vom obține doar o creștere significantă a timpului de calcul a obiectelor iar calitatea suprafețelor rămâne neschimbată. Evedent că în cazul în care gene-

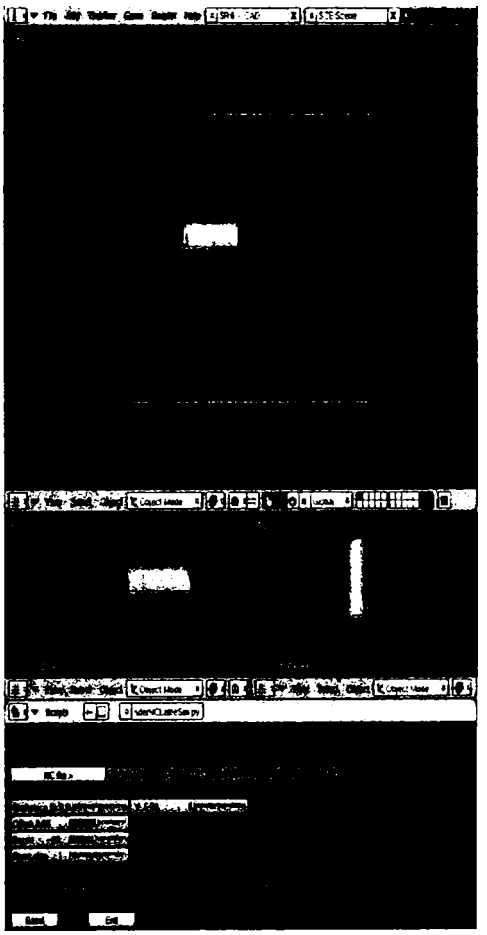


Fig. 239: Configurație cu trei vederi

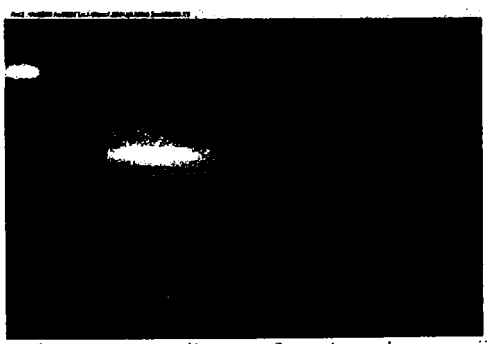


Fig. 241: Prima vizualizare grafică prin render a strunjirii

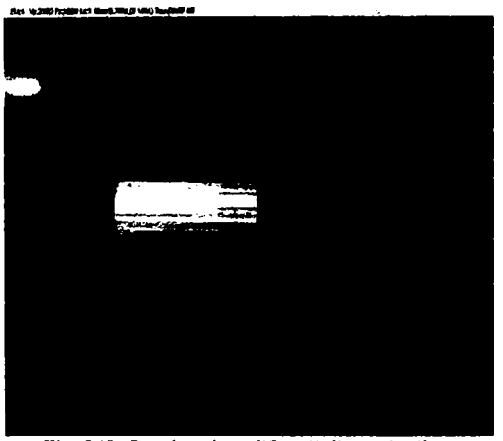


Fig. 240: Rezultatul modificării dimensiunilor semifabricatului

rarea de corpuri cu suprafețe foare mari și/sau a prezentărilor de imagini de rezoluție ridicată (A0), este inevitabil necesar ca acest număr să fie modificat – modificare care însă necesită corecția limitelor incluse în program.

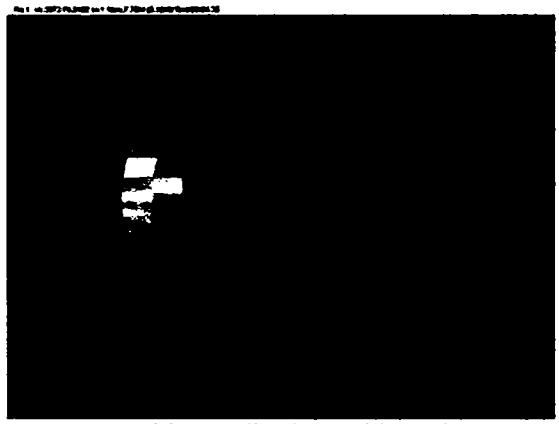


Fig. 242: Modificarea offsetului semifabricatului mașinii

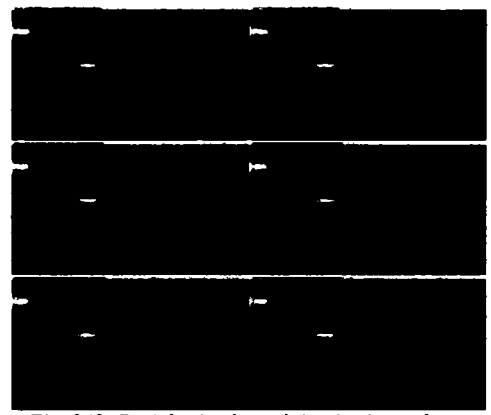


Fig. 243: Pasi de simulare obținuți prin render

În cazul în care doar reprezentarea suprafețelor corpurilor intermediare este dorită (pentru moment delimitatorul utilizat fiind dat de codurile G0, M0 sau M30), este necesară utilizarea următorului 'slider'. Acesta, ca de altfel toate elementele de acest tip, poate fi evident direct editat sau controlat prin mouse clicks, dar pentru ușurarea controlului simulării utilizarea cursorilor tastaturii a fost de asemenea implementată. În cazul în care cursorul mouse-ului este aflat în cadrul suprafeței de GUI control, cursorii vor avea următoarele semificații:

JOS – ultimul pas

SUS – primul pas

STÂNGA – un pas înapoi

DREAPTA – un pas înainte

Secvența pașilor simulării este prezentată în imaginea alăturată.

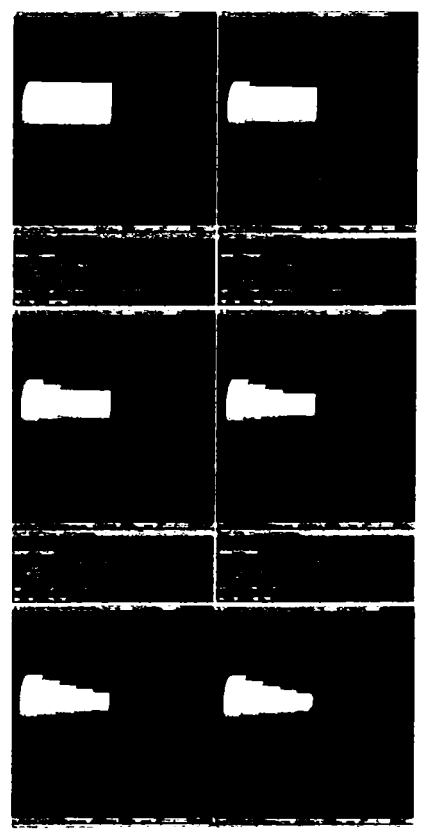


Fig. 244: Pași de simulare

În mod mormal, cea mai mare parte a timpului dedicat simulărilor nu va fi datorată procesului de rendering, deci de generare a unor imagin de calitate cât mai ridicată, ci analizării pas cu pas a programului G-Code introdus prin fișierul de intrare. Din acest motiv, atât dimensiunea semifabricatului cât și poziția planului de offset vor rămâne neschimbate, ceea ce ne permite ascunderea lor prin utilizarea butoanelor de tip toggle 'Show workpiece' și respectiv 'Show Offset pl.'. Ca urmare a acestui fapt, transparența nemaifiind necesară, vom putea urmări simularea pas cu pas în interiorul ferestrelor Blender. Imaginea secvențelor prezentate anterior, de data aceasta fără implicarea procesului de rendering, poate fi regăsită în Fig. 244. Ca în cazul imaginilor trecute prin procesul de rendering, ultimul pas al prelucrării n-a mai fost prezentat, acesta fiind o simplă deplasare rapidă G0, și deci neinfluiențând forma obiectului obținut.

Deplasările sunt reprezentate în program prin benzi colorate, roşii pentru deplasările rapide şi verzi pentru cele de lucru.

Ultimele doua elemente de control din fereastra GUI sunt cele corespunzătoare resetării sistemului (trecerea la valorile iniţial programate şi prevăzute în program) şi respectiv terminării simulării cu predarea controlului programului principal – Blender. Obiectele obţinute în urma simulării pot fi evident modificate şi prelucrate în continuare utilizând mijloacele de modelare şi/sau animaţie oferite de acesta.

9.3.4 Exemple de implementare în sisteme distribuite

Ca exem lu, vom utiliza detec ia cercurilor i contururilor date prin detectorul Hough ca aplicaţie distribuită. În principal putem privi această distribuire a activităţilor din două moduri de aplicare complet diferite. Prima modalitate este referitoare la detectarea obiectelor de dimensiuni sau orientări diferite în cadrul aceleiaşi imagini digitale, iar ce-a de-a doua este privind segmentarea imaginii iniţiale în imagini de dimensiuni mai reduse, analiza distribuită având loc pe baza imaginilor parţiale astfel determinate.

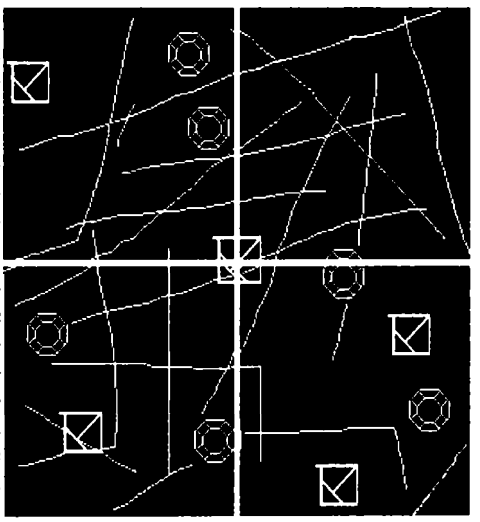


Fig. 245: Segmentarea imaginii în imagini parţiale

Dat fiind faptul că separarea imaginilor în imagini parţiale care să asigure detectarea tuturor contururilor dorite este departe de a fi trivială. În special acest lucru este datorat necesităţii implementării unui sistem de preprocesare care să analizeze şi în primul rând să segmenteze imaginea iniţială în aşa fel, încât contururile dorite să fie cu siguranţă conţinute în cel puţin una din imaginile parţiale segmentate. În lucrarea de faţă ne vom ocupa din acest m tiv impl m r pr gm i impl i i rib iri cercurilor cu diametre diferite şi a detectării imaginilor de referinţă în imagini binare.

În vederea unei clarificări schematice simple a afirmaţiilor an erioare, în imaginea urm oare es e prezen a o imagine iniţială segmentată în două trepte uniforme. După cum se poate observa, în urma acestei împărţiri, deşi simplă şi fără a necesita o analiză a priori a imaginii, nu toate elementele vor fi determinate în urma distribuirii detectării, unele fiind distribuite pe două şi chiar patru imagini separate.

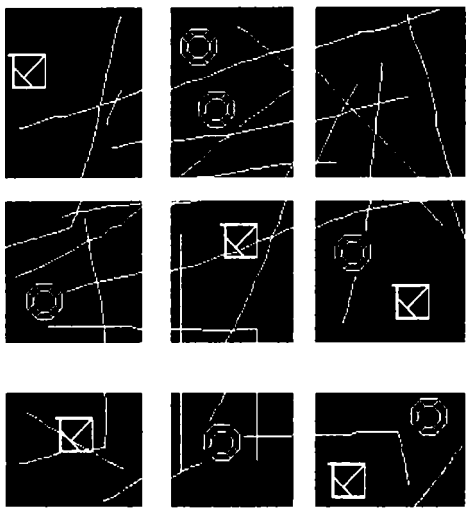


Fig. 246: Segmentarea manuală a imaginii în imagini parţiale

Problema poate fi rezolvată în două moduri, cel mai simplu fiind cel interactiv, dar care în cele mai multe cazuri nu va conduce la obţinerea tuturor rezultatelor dorite, rasterul având posibilitatea să treacă printr-unul din elementele căutate. În cazul imaginilor simple, cum este cea prezentată anterior, modificând manual poziţia şi dimensiunile rasterului, vom obţine o segmentarea asemănătoare celei din Fig. 246. Evident că în acest caz, detectarea corectă a poziţiei contururilor căutate devine trivială, metoda având însă două dezavantaje majore:

- me oda nu poa e fi ut m iz ă i
- timpul necesar operatorului în vederea segmentării imaginilor complicate creşte cu complexitatea imagi-

nii

- poziția de start este diferită pentru fiecare dintre subimaginile astfel obținute, simplul index al poziției subimaginii nefiind suficient în vederea consolidării rezultatelor obținute din detectarea distribuită
- intervenția manuală poate aduce cu sine riscul unei segmentări eronate

Din acest motiv, algoritmul următor propune utilizarea unei metode de segmentarea automată a imaginii. Deși modul de secționare nu este optimizat în ceea ce privește numărul de segmente, implementarea sa este simplă și scalabilă pentru imagini oricât de mari.

Ținând cont de faptul că dimensiunile imaginilor implicate sunt cunoscute iar poziția elementelor de detectat ramânând neschimbată pe timpul execuției, este suficient ca imaginea să fie segmentată în trei moduri diferite, printr-un număr de segmente egale (în cazul de față în două și trei tronsoane de imagini). Evident că în vederea segmentării este necesară utilizarea a de divizori primi între ei, pentru a evita suprapuneri nedorite ale segmentelor. Scriptul utilizat în vederea segmentării automate a imaginii este cel prezentat în anexă sub numele de **autosplit235.sh** – datorită faptului că aceste utilizează segmentarea în două, trei și respectiv cinci tronsoane, iar cele trei seturi de imagini segmentate sunt prezentate în imaginile ce urmează (Fig. 247, Fig. 249 și Fig. 248).

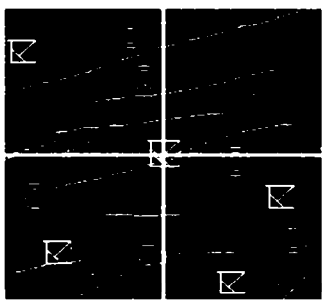


Fig. 247: Segmentarea automată în două tronsoane

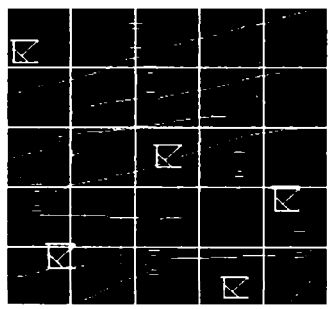


Fig. 248: Segmentarea automată în cinci tronsoane

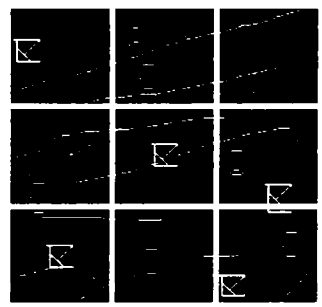


Fig. 249: Segmentarea automată în trei tronsoane

În principiu, datorită dimensiunii relativ reduse a imaginii de intrare, această segmentare este cu siguranță suficientă și pentru imagini de dimensiuni ridicate, timpul necesar detectării elementelor fiind evident triplu față de cel necesar detectării elementelor într-o singură imagine, dar metoda propusă este practicabilă și necesară având în vedere următoarele considerații:

- datorită pe de o parte faptului că pentru imagini de dimensiuni mari detectarea devine imposibilă (în special datorită dimensiunii acumulatorului)
- datorită distribuirii calculelor pe cluster sau calculatoare diferite, metoda oferă posibilitatea obținerii rezultatelor dorite indiferent de dimensiunea imaginii inițiale.

Aplicarea și demonstrația practică a validității metodei va fi prezentată în continuare, scriptele și aplicațiile necesare fiind cuprinse în anexă.

9.3.4.1 Detectarea cercurilor

În vederea detectării cercurilor vom utiliza detectorul generalizat Hough pentru cercuri propus în lucrarea de față, distribuția având loc în funcție de diametrele căutate. Un alt exemplu de utilizare ar fi detectarea cercurilor într-un anumit domeniu de valori pentru imagini deosebit de mari, care din motivele prezentate anterior, necesită divizarea lor, determinarea centrelor cercurilor și recentralizarea rezultatelor. Exemplul utilizat, obținut prin generarea de cercuri în poziții aleatoare într-o imagine de dimensiune 1200x800. Evident că, datorită dimensiunii reduse a diametrului cercurilor de detectat, o segmentare in 2,3 și 5 trepte este suficientă, pentru raze mai mari fiind eventual necesară utilizarea unei trepte suplimentare (7).

Rezultatul segmentării automate este prezentat în imaginile din tabelul următor.

Tab. 15: Segmentarea în vederea detectării distribuite a cercurilor

9.3.4.2 Detectarea formelor complexe

Utilizând detectorul Hough generalizat, este posibilă adaptarea programului în vederea găsirii poziţiei punctului 'hot spot' în interiorul imaginii date, fie pentru variaţii de dimensiune a măştii utilizate, fie în cazul poziţiei (rotaţiei) acesteia. Domeniile de utilizare a rezultatelor pot fi încadrate în:

– utilizarea subprogramelor de comandă numerică prin intermediul unui postprocesor responsabil de adaptarea acestor subprograme (scalarea datelor)

- poziționarea automată a axei boșelor pe mașini de acest tip cu comandâ numerică
- poziționarea relativă a mașinilor de tăiat și conturat cu comandă numerică în vederea prelucrărilor prin prinderi intermediare

Deși utilizarea de subprograme este în versiunea actuală EMC2 realizată, pentru păstrarea generalității și prin utilizarea unui script python foarte simplu, generarea programului complet este relativ simplă. În acest scop, subprogramul de conturat va fi parametrizat prin utilizarea de variabile, acestea fiind succesiv adaptate în funcție de poziția determinată a offsetului din imaginea digitală. În cazul exemplului utilizat (Fig. 250 în care contururile detectate sunt marcate), acest subprogram – de asemenea generat automat din analiza imaginii de referință (**FollowContourRef.java**) este prezentat în continuare.

```
(start subprogram)
(Contour 1)
g0 x[#1001 + 0.200] y[#1002 + 0.200] z0.000
g1 x[#1001 + 0.100] y[#1002 + 0.100] f100.000
g1 z-0.100
g1 x[#1001 + 0.100] y[#1002 + 0.100]
g1 x[#1001 + 2.800] y[#1002 + 0.100]
g1 x[#1001 + 2.800] y[#1002 + 0.200]
g1 x[#1001 + 2.800] y[#1002 + 2.500]
g1 x[#1001 + 2.700] y[#1002 + 2.500]
g1 x[#1001 + 0.400] y[#1002 + 2.500]
g1 x[#1001 + 0.400] y[#1002 + 2.400]
g1 x[#1001 + 0.400] y[#1002 + 0.200]
g1 x[#1001 + 0.500] y[#1002 + 0.200]
g1 x[#1001 + 2.700] y[#1002 + 0.200]
g1 x[#1001 + 2.700] y[#1002 + 0.300]
g1 x[#1001 + 2.700] y[#1002 + 0.300]
g1 x[#1001 + 2.600] y[#1002 + 0.400]
g1 x[#1001 + 1.200] y[#1002 + 1.800]
g1 x[#1001 + 1.100] y[#1002 + 1.700]
g1 x[#1001 + 0.500] y[#1002 + 1.100]
g1 x[#1001 + 0.500] y[#1002 + 1.000]
g1 x[#1001 + 0.500] y[#1002 + 0.300]
g1 z0.100
(Contour 2)
g0 x[#1001 + 0.200] y[#1002 + 0.300] z0.000
g1 x[#1001 + 0.100] y[#1002 + 0.200] f100.000
g1 z-0.100
g1 x[#1001 + 0.100] y[#1002 + 0.200]
g1 x[#1001 + 0.300] y[#1002 + 0.200]
g1 z0.100
(Contour 3)
g0 x[#1001 + 2.700] y[#1002 + 0.400] z0.000
g1 x[#1001 + 2.600] y[#1002 + 0.300] f100.000
g1 z-0.100
g1 x[#1001 + 2.600] y[#1002 + 0.300]
g1 x[#1001 + 2.600] y[#1002 + 0.300]
g1 x[#1001 + 2.500] y[#1002 + 0.400]
g1 x[#1001 + 1.200] y[#1002 + 1.700]
g1 x[#1001 + 1.100] y[#1002 + 1.600]
g1 x[#1001 + 0.600] y[#1002 + 1.100]
g1 x[#1001 + 0.500] y[#1002 + 1.200]
```

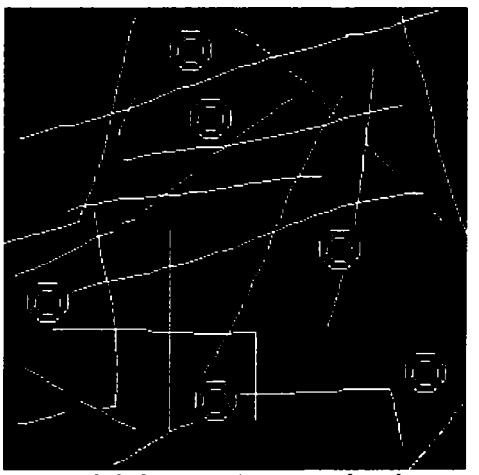


Fig. 250: Detectarea/conturarea formelor complexe - exemplu

```
g1 x[#1001 + 0.500] y[#1002 + 1.200]
g1 x[#1001 + 0.500] y[#1002 + 1.300]
g1 x[#1001 + 0.500] y[#1002 + 2.400]
g1 x[#1001 + 0.600] y[#1002 + 2.400]
g1 x[#1001 + 2.700] y[#1002 + 2.400]
g1 z0.100
(Contour 4)
g0 x[#1001 + 1.400] y[#1002 + 1.900] z0.000
g1 x[#1001 + 1.300] y[#1002 + 1.800] f100.000
g1 z-0.100
g1 x[#1001 + 1.300] y[#1002 + 1.800]
g1 x[#1001 + 1.300] y[#1002 + 1.800]
g1 x[#1001 + 1.400] y[#1002 + 1.900]
g1 x[#1001 + 1.800] y[#1002 + 2.300]
g1 x[#1001 + 1.700] y[#1002 + 2.300]
g1 x[#1001 + 1.700] y[#1002 + 2.300]
g1 x[#1001 + 1.600] y[#1002 + 2.200]
g1 x[#1001 + 1.300] y[#1002 + 1.900]
g1 z0.100
```

După cum se poate observa, subprogramul utilizează variabilele #1001 şi #1002 pentru x şi respectiv y, programul de generare având singurul scop de a seta aceste variabile înainte de includerea codului în programul general.

În vederea generării variantei cu subprograme, sursa anterioară va necesita o mică modificare prin adăugarea liniei de declaraţie şi a celei de sfârşit de subprogram, sub forma

```
(start subprogram)
o100 sub
(Contur 1)
...
g1 z0.100
o100 endsub
```

Evident că apelarea având loc doar prin inserarea liniei

```
o100 call
```

în programul final, şi nu prin inserarea completă a codului subprogramului, acesta va avea dimensiuni mult reduse. Pe de altă parte însă, utilizarea subprogramelor funcţionează doar în ultima versiune EMC2 şi doar sub Ubuntu-Linux, fapt pentru care ambele posibilităţi au fost prezentate.

Apelarea scriptului de distribuţie se va face pe lina de comandă, ca în exemplul următor:

```
./distrib.sh GHM2Distrib GHMRef png 0.1
```

Respectiv

```
./distrib_sub.sh GHM2Distrib GHMRef png 0.1
```

După cum se observă, apelarea scriptului se va face neapărat în directoarea în care vor fi generate de compilatorul Java clasele de execuţie (pentru Eclipse sub <project/bin>).

Parametrii de apelare sunt imaginea iniţială (fără extensie), imaginea de referinţă (fără extensie) extensia ambelor fişiere (va fi utilizată atât pentru segmentele de imagini generate cât şi pentru fişierele imagine generate în vederea documentării şi depanării) , precum şi un factor de scalare pentru trecerea de la pixeli la unităţi reale (mm sau inch) în vederea prelucrării pe maşina cu comandă numerică.

Esențială pentru rularea cu succes a scriptului este existența nu numai a imaginilor digitale de intrare și referință, ci și a subprogramului anterior (conținând comenzile G-Code de generare a curbei de referință). Acest fișier trebuie să poate numele **GHMRef_param.ngc**, deoarece acesta va fi utilizat în vederea generării programului NC final. Deși generarea automată a acestui modul ar putea fi de asemenea inclusă în script prin apelarea modulului amintit anterior, utilizarea unui postfix-ului **_param** permite utilizarea unei versiuni optimizate sau chiar a unui alt subprogram ce va fi rulat în poziția în care obiectele vor fi detectate.

În vederea ușurării apelării programelor în cazul distribuirii, un script corespunzător este automat generat, scriptul **distrib.sh** așteptând rezultatul analizei distribuite în vederea consolidării rezultatelor, continuarea procesării având loc în urma apăsării tastei **Return** sau **Enter**. Numele scriptului generat este derivat din numele fișierului imagine de intrare, având însă extensia **.dist.sh**. Acesta, împreună cu imaginile segmentate dorite putând fi apoi transferate (manual sau tot pe baza unor scripte) pe sistemele de calcul distribuite. În cazul exemplului de mai sus, acesta va avea deci numele **GHM2Distrib.dist.sh**. Dat fiind faptul că toate fișierele generate, fie de tip imagine digitală, fie de tip text sunt derivate de la numele fișierului de intrare, transferul lor între sistemele de calcul este trivial (ftp, sftp, scp, cu comprimare sau fără). În cadrul lucrării de față s-a recurs la distribuirea datelor de intrare și a scriptelor aferent prin intermediul sistemului UNICORE, acesta permițând utilizarea în GRID a sistemului implementat.

De asemenea, scriptul asigura executarea programului de detectare doar în cazul în care segmentul imaginii a fost transferat pe sistemul actual de calcul.

Prin startarea pe sistemul client de calcul a acestui script, toate imaginile corespunzătoare aflate în fișierul director curent vor fi analizate iar rezultatele vor fi salvate în fișiere intermediare, după același principiu.

În urma transferului rezultatelor (fișiere de tip **<fișier de intrare>.dr**) din nou pe sistemul pe care rulează încă scriptul inițial, și după cum am amintit prin apăsarea tastei de **Return** sau **Enter**, scriptul python **ncmontage.py** (respectiv **ncmontage_sub.py**) va fi lansat în vederea consolidării rezultatelor și generării programului G-Code final.

În afara parametrilor amintiți, scriptul de distribuire mai poate fi apelat cu un parametru obțional (a cărui valoare este arbitrară), simpla sa existență generând un script suplimentar de regenerare a imaginii inițiale cu poziționarea contururilor determinate – facilitate foarte utilă în special în vederea documentării și depanării sistemului. Imaginile astfel obținute, în cazul exemplului utilizat, sunt prezentate în Fig. 251, Fig. 252 și Fig. 253.

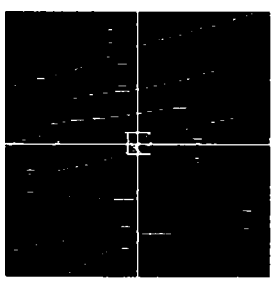


Fig. 251: Segmentarea în vederea distribuirii (2x2)

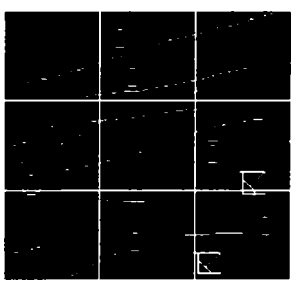


Fig. 252: Segmentarea în vederea distribuirii (3x3)

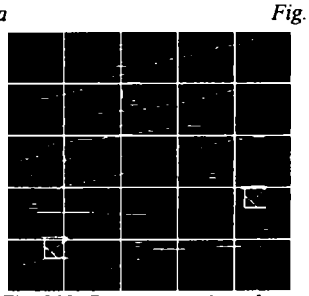


Fig. 253: Segmentarea în vederea distribuirii (5x5)

După cum se poate observa, nici una din segmentări nu poate detecta toate contururile dorite, în schimb, prin utilizarea distribuirii, detectarea independentă a contururilor dorite în fiecare din imaginile parțiale și cumularea rezultatelor, duce la obținerea rezultatului dorit, toate elementele fiind regăsite în reuniunea rezultatelor parțiale.

Rezultatul simulării conturării prin rularea programului NC generat din detectarea distribuită a imaginii digitale considerate este următorul.

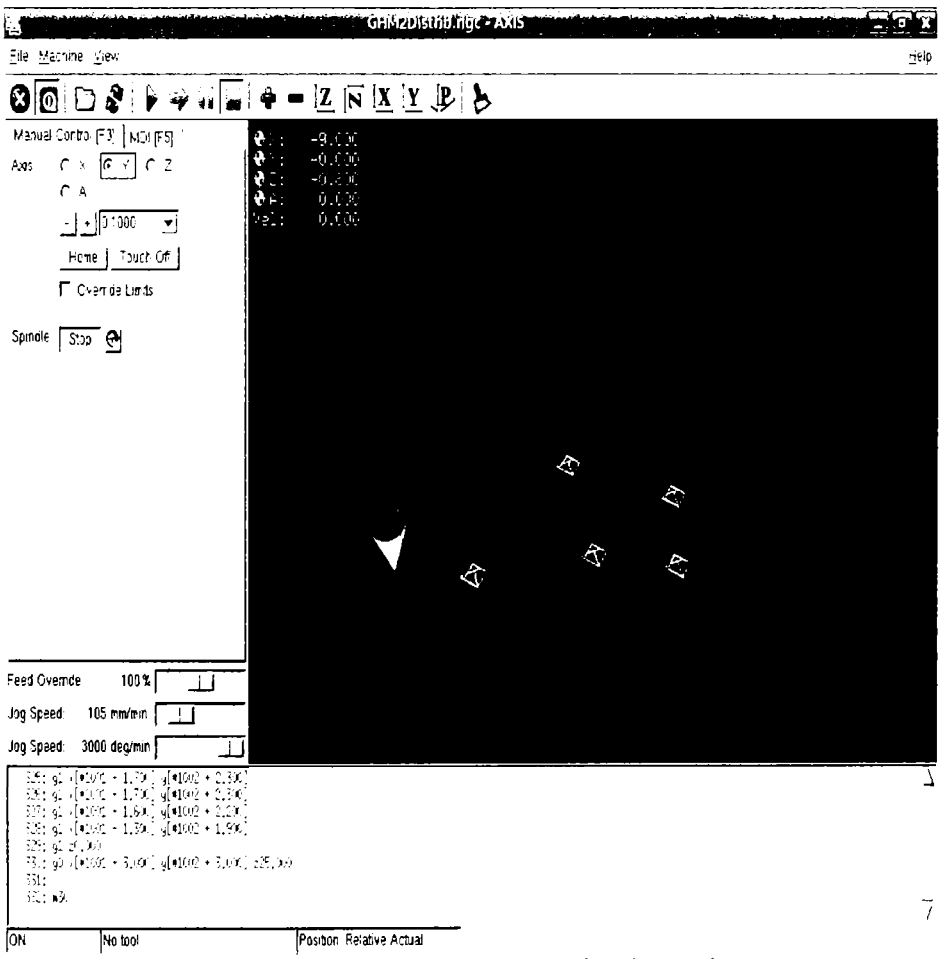


Fig. 254: Rezultatul simulării conturării formelor complexe

10 Platforma de testare

Din punctul de vedere al dezvoltatorilor de programe OS, al studenților și micilor producători, n-ar fi un câștig prea mare, dacă mașinile CNC necesare ar fi scumpe. Doar simularea prelucrărilor, fără posibilitatea de verificare practică a rezultatelor, ar duce la un grad de incertitudine mult prea ridicat și neadecvat. Din acest motiv, unul dintre cei mai importanți pași a fost cel făcut atât de unele firme producătoare de componente electronice și utilaje necesare mașinilor cu comenzi numerice, cât și în principal grupurilor de programatori și utilizatori dornici de obținerea și realizarea unor componente utilizabile în special în scopuri private și didactice (chiar dacă precizia inițială atinsă a lăsat mult timp de dorit). Deși numărul producătorilor de astfel de utilaje este mare și în continuă creștere și în ciuda faptului că mulți producători se pot mândri cu mașini de prototipare rapidă foarte precise și flexibile, în cadrul lucrării de față a fost utilizată exclusiv linia UNI-Turn și UNI-Mill al firmei Sherline [58]/ TheCoolTool[59]. Ambele mașini sunt livrate sub formă de chit de asamblare, și în ciuda acestui fapt, realizează precizii de prelucrare sub sutime de milimetru, iar asamblarea lor nu durează mai mult de câteva ore.

Pe de altă parte, este de amintit faptul că în ciuda prețului accesibil oricărui mașinist, indiferent de aplicabilitatea în domenii de hobby sau profesionale, numărul accesoriilor existente este de asemenea remarcabil. Datorită precizie lor remarcabile și în special a prețului redus, aceste produse sunt de-a dreptul predestinate prototipării rapide prin prelevarea de material, singura restricție fiind dimensiunea redusă a semifabricatelor prelucrabile.

10.1 Lathe – UNI-Turn 2000

Ca majoritatea strungurilor existente, UNI-Turn este programabil în două axe. Pricipalele diferențe față de mașinile 'consacrate' în ceea ce privește [illegible] prin comandă numerică sunt:

- motorul principal nu este controlabil prin CNC, și deci viteza de rotație a axei principale nu poate fi programată – implicit, comanda M3 (rotația în sens orar) este singura posibilă
- utilizarea răcirii nu este controlată prin CNC
- nu există punct de referință al mașinii
- coordonarea avansurilor și a vitezei de rotație a axei principale în vederea filetării prin CNC (G33) nu este posibilă (filetarea manuală prin utilizarea unui dispozitiv adaptor este însă posibilă)

Fig. 255: UNI-Turn 2000

În ciuda acestor impedimente și mai ales datorită prețului și precizie sale de prelucrare, poate fi considarată (cel puțin în momentul de față) ca fiind cea mai adecvată mașină în scopuri educative și de hobby.

10.2 Freza – UNI-Mill 5410

Echipată cu același tip de controler ca și UNI-Turn, mașina corespondentă utilizată pentru frezare este UNI-Mill. Numărul maxim de axe acceptat este de patru, notate în mod standard X, Y, Z și A ca axa de rotație. Exceptând problematica dimensiunilor reduse și a celorlalte impedimente prezentate în cazul UNI-

Turn, maşina permite frezarea de aluminiu şi alamă, plasându-se astfel în momentul de faţă în topul maşinilor comparative de prototipare rapidă existente, care pot prelucra astfel de materiale, datorită faptului că toate celelalte (din aceaşi categorie de parametri) pot prelucra doar răşini sintetice, lemn, plastic sau alte materiale moi.

Versiunea în patru axe utilizată în cadrul lucrării de faţă este prezentată în Fig. 256.

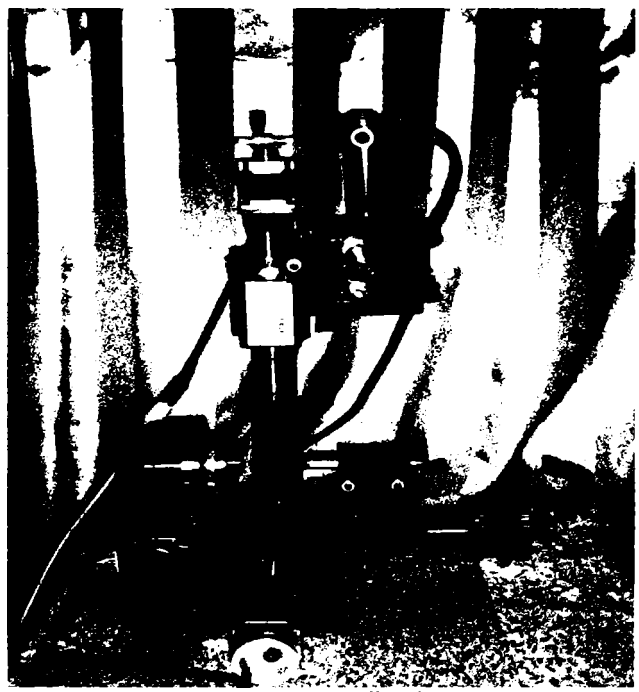


Fig. 256: UNIMill5410

Ambele maşini utilizează un motor pas cu pas de curent continuu, cu echipat cu protecţie termică. Viteza de rotaţie a axei principale poate fi modificată doar manual, într-un domeniu cuprins între 70 şi 2800 RPM, aşadar mai mult decât suficient pentru majoritatea prelucrărilor de strunjire şi frezare realizabile, ţinând cont de dimensiunile reduse ale semifabricatului.

10.3 CNC Software - RT-Linux & EMC2

Însă ea mai importantă parte a prototipării rapide prin prelevare de material însă este partea de soft CNC, căci fără aceasta, prelucrarea unei piese modelate, vizualizate şi simulate - chiar în corelaţie cu un hard relativ ieftin - nu poate fi realizată. Datorită lipsei facilităţilor de operare în timp real (RT) a celor mai multe sistem de operare ale calculatoarelor personale, hardul CNC este de obicei interfaţat printr-un controler hard (firmware), care în general conduce la ridicarea preţului de achiziţie a sistemului. Din acest motiv, în lucrarea de faţă am recurs la utilizarea sistemelor de operare Linux în timp real (iniţial BDI-Linux şi Puppy-Linux, ultimele încercări fiind efectuate sub Ubuntu 6.06). În acest caz, dezvoltarea de programe şi partea de CNC-Controler rezidă în cadrul sistemului de operare.

În anul 2002, NIST a elaborat prima versiune EMC (Enhanced Machine Controller), definindu-l în draftul iniţial al documentului „EMC-Handbook“ [62] (în traducere liberă) astfel:

„Programul Enhanced Machine Controller (EMC) este un effort al NIST în dezvoltarea şi validarea unei specificaţii a interfeţlor controlerelor de arhitectura deschisa. În termeni simpli, EMC este un program CNC-Controler pentru free şi open source. EMC poate controla servo-motoare, motoare pas cu pas, relee şi alte componente legate de maşinile de prelucrare“.

Dialectul EMC este bazat pe NC274 "G-Codes", dezvoltat original de Allen şi Bradley ca parte a muncii lor desfăşurate la NCMS în cadrul proiectului NGC şi este similar seriilor de maşini cu comanda numerică Fanuc. Primele interfeţe ale HMI sub Puppy Linux şi BDI Linux sunt prezentate în continuare.

Principalul avantaj al distribuţiei Puppy-Linux din pachetul livrat de firma "The Cool Tool" este faptul că

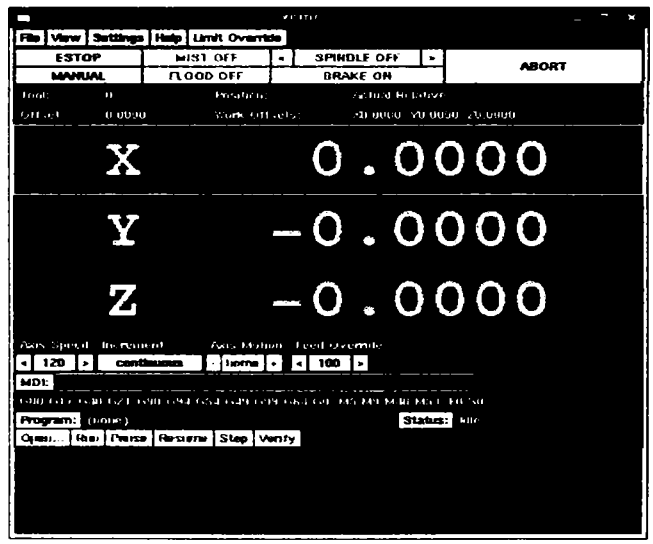
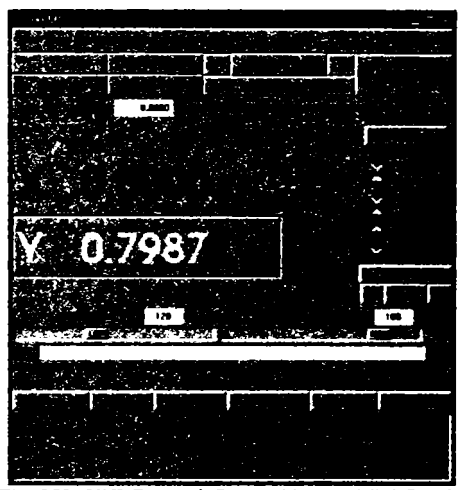


Fig. 257: UNI-Mill cu xemc Display



¯g. ¯9: ´´´´-Turn su´ ¯uppy-´ ´nux ş´ ´n-¯ sp´ay

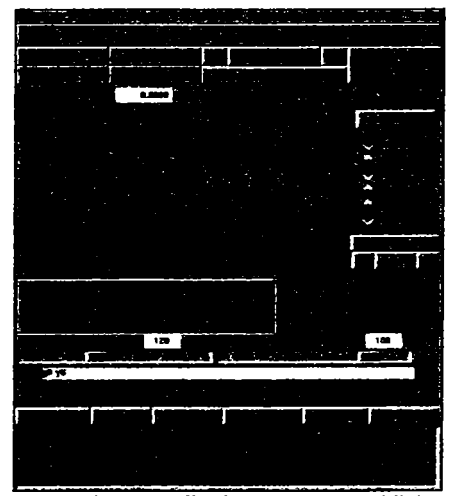


Fig. 258: UNI-Mill sub Puppy-Linux şi Mini-Display

acest sistem poate fi pornit direct de pe CD sau USB-Stick, cerinţele sistemului fiind foarte modeste (ram-disk de 256MB şi CPU > 500MHz), fără a necesita o instalare prealabilă pe disc. Cu toate aceste, în vederea adaptării programelor şi salvarea setărilor intermediare, o instalare 'normală' pe unitatea de disc a PC-ului este recomandabilă.

Versiunea DBI a fost prima versiune mult mai complexă, flexibilă şi având posibilitatea instalării tuturor opţinunilor necesare unui sistem de operare modern, actual şi complet (inclusiv KDE, Graphics, Multimedia, OpenOffice şi aplicaţii ştiinţifice).

Actuala versiune, apărută în 2007, este bazată pe distribuţia Ubuntu 6.06 LTS şi cuprinde toate avantajele

celorlate sisteme prezentate împreună cu o versiune stabilă a unuia dintre cele mai bine puse la punct distribuţii Linux. Pe lângă modulul de Display prezentat anterior (TkEMC), această distribuţie mai cuprinde şi Axis, ale cărui snapshots au fost prezentate în figurile anterioare referitoare la frezarea şi tăierea profilelor şi contururilor detectate din analiza imaginilor binare, în special în vederea comparării lor cu alte sisteme existente comparabile atât din punct de vedere al posibilităţilor lor de simulare şi prelucrare cât şi a preţului (deşi în cazul sistemului de faţă preţul se reduce la eventualele cheltuieli legate de download şi generarea CD-ului de instalare).

11 Utilitare de tip Freeware şi OpenSource utilizate

Luând în considerare ţelul principal al lucrarii de fată, acela de a pune la dispoziţia tuturor prin utilizarea de mijloace în principal Open Source, în vederea implementării şi utilizării utilitarelor necesare prototipării rapide cu scopul optimizării proceselor şi implicit a măririi capacităţii de prelucrare a maşinilor corespunzătoare, vom aminti în continuare principalele sisteme de operare şi utilitare de acest tip utilizate, cu o scurtă explicaţie de rigoare.

Blender - Unul dintre cele mai puternice utilitare[52] din domeniul modelării 3D, animaţiei şi vizualizării. Avantajul principal este interfaţarea cu C++ şi Pyton, permiţând astfel o abordare atât interactivă cât şi programatică a problemei de vizualizat. Datorită posibilităţile pe care le oferă prin înglobarea de module utilitare externe (plug-ins), sistemul este deschis şi adaptabil cerinţlor speciale impuse de problemele ce urmează a fi rezolvate.

Eclipse - IDE (Integrated Development Environment) – unul dintre cele mai puternice IDE existente la ora actuală, Open System cu numeroase plug-ins incorporate sau incorporabile. Utilitarul este utilizabil sub CPL (Common Public Licence) şi gratuit pentru dezvoltarea de programe OpenSource.

Deşi există numeroase sisteme integrate (IDE – Integrated Development Environment) în vederea dezvoltării de programe, unul dintre cele mai convenabile în vederea realizarii scopului propus prin această lucrare este Eclipse[63]. Utilitarul a fost ales în special datorită facilităţilor sale de integrare modulară atât a programelor Java cât şi a celor Python. Deşi, de la caz la caz, au fost utilizate şi alte utilitare de acest tip (IDLE, SPE, vi, Nedit), marea majoritate a implementării a fost efectuată utilizând acest IDE.

EMC2 - Modulele principale necesare părţii de CAM sunt date de EMC (Enhanced Machine Controller). Prima versiune a acestui sistem a apărut în anul 2002, iar la ora acutală este cunoscut sub denumirea de EMC2[62], poate comanda freze şi strunguri cu până la 6 axe.

EDISON – (Edge Detection and Image Segmentation System) Deşi în cazul acestui program[40] este vorba doar de utilizarea surselor algoritmului prezentat, dat fiind faptul că prin bunăvoinţa autorilor sursele au fost puse la dispoziţie şi sunt utilizabile pe toate sistemele existente, amintirea lor împreună cu mulţumirile de rigoare este obligatorie (cel puţin din punct de vedere moral).

GIMP - Atât datorită posibilităţilor sale grafice extrem de puternice, independenţei de sistemul de operare şi a modalităţii de programare interactivă, GIMP[64] a devenit unul dintre cele mai cunoscute şi apreciate programe de prelucrare grafică a imaginilor digitale. Multe din programele implementate în Java şi prezentate în acestă lucrare erau cu un an înainte inexistente în GIMP, iar la ora actuală numărul filtrelor implementate depăşeşte limita oricăror aşteptări. Datorită faptului ca programul poate fi coordonat prin intermediul liniei de comandă, îl promovează în rândul programelor indispensabile pentru prelucrarea (cel putin iniţială) a imaginilor digitale. Unul dintre cele mai mari avantaje însă îl prezintă faptul că prin simpla utilizare a untilitarului gimptool este posibilă integrarea de noi filtre (sau plug-ins) scrise de utilizator în C.

ImageJ - Program în Java, dezvoltat de Wayne Rasband de la U.S National Institute of Health care conţine numeroase din metodele de prelucrare a imaginilor digitale, program utilizat în special în vederea testării algoritmilor implementaţi[65].

ImageMagic - Utilizat în principal în vederea convertirii în batch a imaginilor digitale, programul[66] a fost de asemenea utilizat pentru separarea imaginilor digitale în vederea distribuirii analizei în GRID.

JDK - Java Developer Toolkit – produs al firmei Sun, unul dintre cele mai răspândite interpretoare ale limbajului Java, prin care se asigură independenţa faţă de sistemul de operare sub care programele vor fi rulate.

Modulul de bază utilizat pentru implementarea aplicaţiilor din cadrul lucrării de faţă este Java2D.

OpenOffice - Pachet de programe Office, existente atât pentru Linux cât şi pentru Windows. Deşi versiunea utilizată se află doar în faza de Beta-Release, prin facilităţile oferite (în special a utilizării gratuite şi compatibilităţii cu Pachetul Office al firmei Microsoft), calitatea produsului are relativ puţine opţiuni ce lasă de dorit, şi care cu siguranţă vor fi remediate sau îmbunătăţite în într-una din versiunile următoare.

Python - Deşi unul dintre cele mai noi limbaje de programare existente (1990), Python a cucerit tot mai mult teren în dezvoltarea de programe, în special datorita conceptului implementării sale – pe de o parte uşor de învăţat şi utilizat, iar pe de altă parte permiţând concomitent programarea orientată pe obiecte, cea orientată pe aspecte şi cea funcţională. Datorita acestor facilităţi este tot mai des folosit şi în cadrul multor aplicaţii ca interpretor în vederea programării de facilităţi suplimentare.

RT-Linux - Deşi toate programele implementate sunt independente de sistemul de operare utilizat, ele au fost concepute şi implementate sub sisteme Linux în timp real (RT-Linux). Principalele distributii utilizate sunt BDI-4.3 şi Ubuntu 6.06, datorită faptului că aceste sisteme au modulele necesare de EMC instalate.

SciLab - Unul dintre cele mai puternice programe matematice de pe piaţă, utilizabil în mod liber cu excepţia elaborării de produse comerciale, caz în care posesia unei licenţe speciale autorizate de INRIA şi ENPC este necesară.

SUSAN - Motivil amintirii acetui packet este similar lui EDISON. Cu toate că utilizarea segmentării în cazul lucrării de faţă este neesenţială, pachetul va avea cu siguranţă – în special datorită vitezei sale de convergenţă a rezultatelor – un rol deosebit în aplicaţii ulterioare.

Umbrello - Utilitar pentru modelarea în UML utilizat atât pentru documentarea exemplelor prezentate în această lucrare, cât şi (în mare parte) pentru prototiparea rapidă a claselor utilizate.

UNICORE 6.0 - Deşi în versiunea 6.0, sistemul de utilitare UNICORE este un sistem relativ nou, implementat aproape complet in Java, apărut datorită necesităţii distribuirii aplicaţiilor în GRID. Împreună cu GridSphere, un alt sistem OpenSource care implementează portalele WEB necesare, UNICORE a reuşit la ora actulă să depăşească (cel puţin din punct de vedere al numărului de instalaţii) majoritatea sitemelor CORBA existente. La ora actuală există numeroase aplicaţii şi Proiecte GRID bazate atât pe acest sistem cât şi pe GlobusToolkit 4.0. În lucrarea de faţă a fost utilizat doar UNICORE, deşi evident, generalitatea ideilor prezentate nu este influenţată de alegerea modului de distribuire a aplicaţiilor - şi deci implicit a utilitarealor folosite. Din acest motiv şi pe acelaşi principiu pot fi folosite de exemplu CORBA sau RMI, scriptele implementate necesitând însa evident modificările de rigoare.

Visualization Toolkit (VTK) - Datorită posibilităţilor sale de vizualizare a datelor 3D şi în special a interfeţei cu C++, Java, Python şi Tcl, VTK este unul din utilitarele cate în ultimii ani au câştigat tot mai mult teren în domeniul vizualizării datelor medical şi ştiinţifice.

12 Concluzii şi contribuţii personale

Luând în considerare amploarea deosebită pe care a luat-o prototiparea rapidă în ultimul timp în foarte multe domenii de activitate, precum şi faptul că maşinile utilizabile în cadrul prototipării rapide sunt de asemenea pe zi ce trece tot mai accesibile, necesitatea optimizării capacităţilor operaţionale a maşinilor de prototipare rapidă (şi în special a celor prin prelevare de material) devine tot mai accentuată.

Evident că o mărire a capacităţilor operaţionale a maşinilor de prelucrare prin prelevare de material este incontestabil realizabilă prin metode clasice, cum ar fi optimizarea parametrilor de aşchiere, creşterea vitezelor de operare, a optimizării profilelor sculelor aşchietoare, optimizarea traseelor sculei aşchietoare etc. Pe de altă parte însă, nereducând prototiparea rapidă doar la procesul de fabricaţie rapidă a prototipurilor şi luând în considerare faptul că aceste tipuri de optimizare sunt în general mult mai adecvate procesului de optimizare al producţiei decât celui de prototipare rapidă, **în cadrul lucrării de faţă am încercat pe de o parte adoptarea, iar pe de altă parte adaptarea, unor metode şi algoritmi utilizaţi în alte domenii, în scopul utilizării lor în prototiparea rapidă prin prelevare de material**. Din acest motiv, am considerat procesul de prototipare rapidă la modul său general, asemănător prototipării rapide existente şi în alte domenii (IT, medicină), ca un proces pornind de la ideea iniţială până la validarea acesteia.

În acest scop, **ideea de prototipare rapidă a fost detaşată de cea a fabricaţiei rapide a prototipurilor**, integrând uneori chiar procese de Reverse Engineering şi Reengineering, tocmai datorită faptului că în cadrul acestui proces se trece de la idee, schiţă, imagine la realizarea iniţial virtuală a produsului finit (Prototipare Virtuală) - iar în urma validării funcţionalităţii acestuia, la realizarea sa fizică. Prin simpla adoptare a acestui principiu, costurile necesare în vederea obţinerii unui prototip sunt net inferioare celor tradiţionale, erorile de concepţie şi cele de integrare (ce pot duce la rebutare de materiale, cheltuieli de producţie şi în special a pierderilor de timp) fiind minimizate, ele fiind aplicate doar în faza finală – de prototipare fizică, reală.

În lucrarea de faţă am plecat de la **ideea demonstrării faptului că, la ora actuală, prototiparea rapidă prin prelevare de material poate deveni accesibilă pe un front foarte larg**, începând de la persoane private (studenţi, hobby-isti, artişti), la mici meseriaşi (bijutieri) şi chiar firme mici. Din acest motiv **a fost decisă utilizarea în exclusivitate utilitare de tip Freeware, Public Domain sau Open Source**, permiţând astfel celor interesaţi dezvoltarea proprie în direcţia dorită a programelor implementate.

Dat fiind faptul că ideea primară este de cele mai multe ori în forma unei schiţe sau imagini (în cadrul artiştilor şi a hobby-istilor acest lucru având loc în general în mod exclusiv), în prima parte a lucrării **au fost prezentate metodele actuale a prelucrării imaginilor digitale, cea mai mare parte a lor fiind implementate în programele Java salvate pe CD-ul anexat (un numar de peste 10 module)**. Deşi în momentul de faţă imaginile necesare prototipării rapide sunt cele binare, în vederea obţinerii lor, algoritmii, modelele şi formatele cele mai cunoscute şi utilizate au fost prezentate. Un alt rol deosebit îl joacă însă imaginile in nuanţe de gri, prin intermediul cărora **am implementat un algoritm de prelucrare prin frezare în 2,5 coordonate**. Pe de altă parte însă algoritmii pot fi cu mare uşurinţă adaptaţi în vederea optimizării procesului de fabricaţie, de exemplu prin reducerea timpilor de poziţionare a plăcilor de conturat/tăiat în funcţie de repere detectabile prin camere video amplasate pe maşina cu comanda numerică. După detectare, adaptarea automată a programului de prelucrare prin aplicarea rotaţiei şi translaţiei faţă de poziţia etalon devine dacă nu trivială, cel putin simplă de implementat.

Pornind de la analiza imaginilor digitale (evident apărute în cadrul scanării sau fotografierii cu camere digitale), **am prezentat nu numai modelele matematice necesare ci am elaborat şi programele aferente în vederea aplicării filtrelor digitale şi detectării contururilor (17 programe numai pentru analiza imaginilor digitale prezentate în anexa1).** Deşi la ora actuală există numeroase utilitare care implementează de asemenea aceşti algoritmi, acestea sunt implementate în diverse limbaje de programare, sunt de cele mai multe ori utilizabile doar în mod interactiv (deci nepermiţând o automatizare a procesului) sau există doar sub anumite sisteme de operare. Prin implementarea lor în Java este asigurată astfel portabilitarea programelor, iar sursele fiind accesibile este permisă modificarea, adaptarea şi dezvoltarea lor.

Ţinând cont de rolul deosebit al simulării în cadrul prototipării rapide virtuale, prin implementarea de pro-

grame aferente, **a fost demonstrat faptul că prototiparea rapidă poate fi susţinută prin mijloace ieftine (de tip Open Source) prin utilizarea utilitarului Blender**. Chiar dacă proiectele propuse sunt încă la început, dezvoltarea lor ulterioară poate fi urmărită în cadrul comunităţii SourceForge. Ca urmare a faptului că în domeniul tehnic majoritatea proceselor de prototipare încep de la modelare, în cadrul unuia din programele elaborate am demonstrat, că **Blender este cu siguranţă adecvat simulării, vizualizării şi generării de G-Code în vederea prototipării rapide** - prin urmărirea curbelor generalizate, simularea frezării, vizualizarea procesului şi generarea de G-Code 'on the fly'. Interfaţa principală de programare interactivă a lui Blender este utilizabilă numai în limbajul python, motiv din care a fost necesară utilizarea acestui limbaj în vederea atingerii scopului dorit. Prin faptul că python este un limbaj independent de platforma de rulare, generalitatea şi scopul lucrării nu sunt afectate ca urmare a adoptării acestei decizii. Interfeţei fiindu-i oferit accesul la totalitatea obiectelor şi funcţiunilor utilitarului, simularea frezării într-un număr arbitrar de axe este de asemenea în viitor posibilă.

Un alt program implementat este cel dedicat strunjirii cu comandă numerică (peste 12 module prezentate în anexa 3). În cadrul acestuia a fost implementată o metodă de simularea prin Blender a prelucrării rezultate din analiza unui fişier NC de intrare. Deşi implementarea este departe de a fi completă în ceea ce priveşte gradul de acoperire a comenzilor standardizate, ea a fost prezentată ca 'proof-of-concept' pentru a demonstra facilităţile şi flexibilitatea utilitarului Blender în vederea simulării interactive şi vizualizării procesului de strunjire.

Atât în vederea simulării cât şi a analizării imaginilor de dimensiuni mari, utilizarea sistemelor distribuite devine inevitabilă. În cadrul lucrării **am elaborat** din acest motiv **un program de detectare distribuită atât a cercurilor cât şi a unor contururi referenţiale, precum şi a programului aferent de generare de G-Code în vederea găuririi - împreună cu simularea corespunzătoare a procesului (19 module prezentate în anexa 4).** Deşi algoritmul utilizat în transformarea Hough generalizată este cunoscut, cel implementat în cadrul lucrării a necesitat o adaptare prin faptul că nu doar maximele acumulatorului au fost utilizate, ci acumulatorul a fost considerat şi el ca fiind o imagine digitală, căreia în urma aplicării unui proces de thresholding, imaginea binară rezultă a fost supusă unui proces de ultimă eroziune (de asemenea modificat prin utilizarea unui buffer suplimentar în vederea stocării poziţiilor centrelor sau punctelor de interes detectate). **Rezultatul astfel obţinut a fost supus unui proces de validare printr-o metodă de template/pattern matching, obţinându-se astfel o stabilitate şi o convergenţă ridicată a algoritmului de detectare.**

Unele metode şi unii algoritmi, deşi foarte importanţi au fost doar amintiţi, implementarea lor fiind relativ dificilă în condiţiile stabilite

Din cercetările efectuate au rezultat următoarele direcţii prioritare de continuare a studiilor:

- Determinarea completă a colţurilor şi liniilor corespunzătoare urmată de generarea de G-Code (combinarea algoritmilor Harris/Hough)
- Simularea utilizând profilul sculelor aşchietoare
- Detectarea rotaţiilor şi tranzaţiilor
- Determinarea arcurilor de cerc în vederea implementării interpolărilor circulare
- CNC Online Design Environment

Index alfabetic

CORBA

Common Object Request Broker Architecture.....195

CSCW

Computer Supported Colaborative Working.....23

CV

Computer Vision.....146

DCAM

Distributed Computer Aided Manufacturing.....9

DFN

Deutsche Forschungsnetz.....157

DLR

Deutsches Zentrum für Luft- und Raumfahrt.....157

EDISON

Edge Detection and Image Segmentation System.....134, 194p.

EEA

Economic Espionage Act.....16

EMC

Enhanced Machine Control.....154, 194p.

FEM

Finite Element Method.....14, 18, 20, 22p., 28

G-code

G-code.....24

HAL

Hardware Abstraction Layer.....155

HPC

High Performance Computing.....22, 156, 159

HSC

High Speed Cutting.....29

IDE

Integrated Development Environment.....194

IGES

International Graphics Exchange Service.....25

LuT

Lookup Table.....69p.

MDI

Machine Device Interface.....154

NC

Numerical Control.....12, 25p., 151, 153

NIST

National Institute for Standards and Technology.....154, 178, 191

NURBS

Non Uniform Rational Binary Splines.....20pp.

PLM

Product Lifecycle Management.....35

PPP

Public Private Partnership.....157

RAPS

Roland Active Piezo Sensor.....24

RE

Reverse Engineering.....14, 20pp.

RMI

Remote Method Invocation.....195

RML

Roland Machine Language....24

RP

Rapid Prototyping....11, 75

RT

Real Time....154p., 191, 195

SDV

Scule Dispozitive Verificatoare....12

Software Engineering

SE....18

SRP

Subtractive Rapid Prototyping....24

STEP

STandard for the Exchange of Product model data....21p., 29, 34, 151

STL

STereo Lithography....21p., 25

SUSAN

Smallest Univalue Segment Assimilating Nucleus....136, 195

TRIPS

Trade-Related Aspects of Intelectual Property Rights....16

UML

Unified Modeling Language....195

USAN

Univalue Segment Assimilation Nucleus....136

VP

Virtual Prototyping....11, 14, 31

VR

Virtual Reality.....9, 12, 14, 18, 23

VRML

Virtual Reality Modeling Language.....23

WTO

World Trade Organization.....16

Index al tabelelor

Bibliografie

1: P.Berce, N.Bâlc, M.Ancău, S.Comşa, C.Caizar, H.Jidav, H.Chezan, Fabricarea rapidă a prototipurilor, 2000

2: Olivier de Weck, David Wallace, Peter Young, Il Yong Kim, Teaching and Education Enhancement Program, Modern Engineering Design and Rapid Prototyping, 2004

3: P.Samuelson, S.Scotchmer, The Law and Economics of Reverse Engineering, 2002

4: E.J.Chikofsky, J.H.Cross II, Reverse Engineering and Design Recovery: A Taxonomy, 1990

5: WTO, Agreement on Trade-Related Aspects of Intellectual Property Rights, 1994

6: US Federal law, Economic Espionage Act, 1996

7: P.Heald, Federal Property Law and the Economics of Preemption, 1991

8: J.E.Cohen, M.A.Lemley, Patent Scope and Innovation in the Software Industry, 2000

9: U.S.C. sec. 1201, There is a limited exception to enable bypassing technical controls and making tools to enable this when necessary to achive interoperability among programs, 1998

10: American Law Institute, Restatement of the Law of Unfair Competition, 1993

11: B.Startly, A.Lau, W.Sun, W.Lau, T.Bradbury, Direct slicing of STEP based NURBS models for layered manufacturing, 2005

12: Todd Grimm, Roland DG Corporate Review, 2003

13: TechnoISEL, Catalog H22T6X-MCX-0, 2004

14: Scott Bury, More than just pretty pictures, 2000

15: MSC-Software, http://www.mscsoftware.com/products, 2005

16: UGS Homepage, http://www.ugs.com

17: W. Abmayr, Einführung in die digitale Bildverarbeitung, 1994

18: W.Burger, M.J.Burge, Digitale Bildverarbeitung, 2005

19: Adobe RGB Color Space Specification (Draft), http://www.adobe.com/support/downloads/main.html

20: R.M. Haralik, K. Shanmugam, I. Dinstein, Textural features for image classification, 1973

21: R.C. Gonzalez, R.E. Woods, Digital Image Processing, 1992

22: C.Iacob, A.Crăciunescu, C.Cristea, L.Dragoş, Ş.Gheorghiţă, R.Trandafir, Matematici clasice şi moderne, 1979

23: R.Resnick, D.Holliday, Physics, 1977

24: An algorithm for the machine calculation of complex Fourier series, 1965

25: W.Abmayr, Einführung in die digitale Bildverarbeitung, 1994

26: E.O.Brigham, FFT Schnelle Fourier Transformation, 1982

27: A fast computational algorithm for discrete cosine transform, 1977

28: P.Zamperoni, Methoden der digitalen Bildbearbeitung, 1989

29: T.Jochems, Transformationen der mathematischen Morphologie - Vorlesungsmitschrift, 1991

30: C.Arcelli, L.Cordella, S.Levadi, Parallel thinning of binary pictures, 1975

31: R.A.Kirsch, Computer determination of the constituent structure of biological images, 1971

32: P.J.Burt, E.H.Andelson, The Laplacian pyramid as a compact image code, 1983

33: P.Zamperoni, Methoden der digitale Bildsignalverarbeitung, 1989

34: J.Lee, R.M.Haralik, L.G.Shapiro, Morphologic edge detection, 1986

35: C.G.Harris, M.Stephens, A combined corner and edge detector, 1988

36: Method adn means of recognizing complex patterns, 1962

37: D.H.Ballard, C.M.Brown, Computer Vision, 1982

38: J.Illingworth, J.Kittler, A Survey of the Hough Transformation, 1988

39: B.Jähne, Practical Book on Image Processinf for Scientific Applications, 1997

40: , http://www.caip.rutgers.edu/riul/research/robust.html

41: Mean Shift Based Clustering in High Dimensions:A Texture Classification Examplen, 2003

42: Efficient Graph-Based Image Segmentation, 2004

43: The SUSAN Principle, http://users.fmrib.ox.ac.uk/~steve/susan/susan/node2.html#SECTION00020000000000000000

44: Efficient Graph-Based Image Segmentation, 2004

45: , Einführung in die digitale Bildbearbeitung, 1994

46: H.Genchi, K.Mori, Evaluation and feature extraction on automatic pattern recognition system, 1965

47: Putting Objects in Perspective, 2006

48: Automatic Photo Pop-Up, 2005

49: Günstiger 3d Laserscanner und schnelle Oberflächenregistrierung, http://www.rob.cs.tu-bs.de/news/david

50: Future Issues for CAD/CAM and Intelligent CNC manufacture,

51: Die Deutsche GRID Initiative (D-Grid), http://www.d-grid.de/

52: Blender Homepage, http://www.blender.org

53: Blender Documentation, 2003

54: Blender for robotics and robotics for Blender, 2005

55: R. Smith, Open Dynamics Engine v0.5 - User Guide, 2004

56: OpenEXR Homepage, http://www.openexr.com

57: Industrial Light & Magic Homepage, http://www.ilm.com

58: Sherline Homepage, http://www.sherline.com

59: TheCoolTool Homepage, http://www.thecooltool.com

60: EMC2 Homepage, http://www.linuxcnc.org

61: NIST Homepage,

http://www.isd.mel.nist.gov/personnel/kramer/pubs/RS274NGC_3.web/RS274NGC_3TOC.html

62: EMC2 Homepage, http://www.linuxcnc.org/

63: Eclipse Homepage, http://www.eclipse.org

64: GIMP Homepage, http://www.gimp.org/

65: ImageJ Homepage, http://rsb.info.nih.gov/ij/

66: Image Magic Homepage, http://www.imagemagic.org

67: M.I.Tamaş, T. Iclănzan, New Methods in Rapid Prototyping using Open Source, Academic Journal of Manufacturing Engineering, vol. 4, nr. 4, 2006, p. 52-61, ISSN 1583-7904

68: M.I.Tamaş, Automatic G-Code Generation and NC-Drill Simulation from Contours Detected in Binary Images, Academic Journal of Manufacturing Engineering, vol. 5, nr. 4, 2007, p. 65-73, ISSN 1583-7904

69: M.I.Tamaş, T.Iclănzan, Using Blender for Lathe-Turning NC/CNC G-Code Simulations, International Journal of Advanced Manufacturing Technology (submitted nov. 2007)

Titluri recent publicate în colecția „TEZE DE DOCTORAT" seria 8: Inginerie Industrială

1. **Liliana Daniela Moşteoru** – *Contribuţii la îmbunătăţirea performanţelor termice şi la ecologizarea aparatelor de sterilizare a instrumentelor medic*ale , *ISBN 978-973-625-441-3*, (2007);

2. **Mariana Ilie** – *Etude de l'interaction laser matière dans le cas des polymères semi-transparents: applications au soudage des polimères, ISBN 978-973-625-449-9,* (2007);

3. **Puiu Câneparu** – *Contribuţii privind îmbunătăţirea performanţelor echipamentelor mecanizate de tăiere termică, ISBN 978-973-625-478-9,* (2007);

4. **Marius Cătălin Grămescu** – *Aspecte privind strategii de dezvoltare specifice intreprinderilor mici şi mijlocii în vederea alinierii la cerinţele Uniunii Europene, ISBN 978-973-625-486-4,* (2007);

5. **Corina-Dana June** – *Optimizarea procesului de încărcare prin sudare în mediu de gaz protector cu rată mare de depunere – încărcarea prin sudare mag cu electrod bandă, ISBN 978-973-625-501-4,* (2007);

6. **Gheorghe Marcel Mocuţa** – *Contribuţii la reducerea intensităţii energetice în judeţul Bihor, ISBN 978-973-625-517-5,* (2007).

EDITURA POLITEHNICA