BILDBEHANDLING
Kort inledning för datorgrafiker

Magnus Bondesson
Institutionen för Datavetenskap
1990-01-12 (Mac), 96-12-13 (UNIX, kraftigt omarbetat), 97-02-23 (MATLAB 5.1), 00-02-21 (red), 01-02-16 (red+MATLAB 6), 02-02-25(2 fel), 04-02-25(MATLAB 6.5)

1 Inledning

Bildbehandling handlar om hur man med dator förbättrar bilder eller utläser information ur bilder. Det kan röra sig om medicinska bilder (t ex röntgenbilder, mikroskopbilder), satellitbilder, bilder av produkter som skall avsynas före slutligt godkännande o s v. Förbättringarna kan avse t ex brusreducering och/eller korrigering för geometriska fel.

Bildbehandlingssystem var länge mycket dyra, t ex kring 1 miljon. Det var först i och med persondatorernas intåg man kunde åstadkomma något till lägre kostnad. Både till PC-familjen och Macintosh finns nu bra bildbehandlingssystem, såväl kommersiella som "public domain".

Bildbehandling har mycket gemensamt med datorgrafik. Den här pappersbunten ger skriftligt underlag för den föreläsning om bildbehandling som brukar ingå i datorgrafikkursen. Självklart ger dessa ord och den tillhörande praktiska demonstrationen bara en glimt av området. En bild kan precis som ljud ses som en signal, dvs alla de hjälpmedel som hör till signalbehandling, är aktuella. Det bör också betonas att i den mån algoritmer och programexempel finns med så är de förenklade och inte speciellt effektiva. Ta alltså vad som sägs här bara som ett inledande budskap från ett spännande område.

I modern bildbehandling spelar **fourieranalys** och **wavelets** (sv krusningar) en betydande roll. Vi berör fourieranalys ytterligt kortfattat, men kan inte gå in på wavelets.

2 Exempel på användningar

På något vis är det väl så att medicinska tillämpningar är mest tilltalande. Här finns också många situationer där goda bildbehandlingssystem kan vara av värde. Tidigt fanns system för räkning av vita blodkroppar i blodprov. Många andra exempel på automatiserad mikroskopi finns. Ett sjukhusbesök kan ofta innebära en mängd provtagningar. Röntgenverksamheten är på väg att helt datoriseras, ett hinder har hittills varit det enorma minnesutrymme som en röntgenbild upptar.

Rymdbilder (dvs bilder tagna från satellit mot t ex jorden) behöver dels korrigeras geometriskt, dels bearbetas på andra sätt. I Kiruna finns företaget Metria Satellus (ingår i Lantmäteriverket; f d Satellus; f d Satellitbild) som ägnar sig åt sådan verksamhet. Man tar emot bilder från satellitserien Landsat (upplösning 30 m; gammal men inte avsomnad) och från den franska SPOT-familjen (upplösning 10-20 m) och senare EROS-serien (upplösning neråt 1 m) och säljer efter bearbetning dem vidare. Man kan t ex producera kartor över nya kalhyggen och har gjort bildatlaser för utvecklingsländer utan egen karttradition. Rymdbilder och bilder från flygplan används för fjärranalys, som avser att ge information om olika fysiska resurser på jorden. Via `http://www.terraserver.com` kan man för vissa delar av världen titta på svart-vita bilder med en upplösning om nedåt 1 m/bildpunkt tagna från en rysk satellit 1989 resp med amerikansk flygfotografering 1993. Adressen `http://www.spacepix.com` har färgbilder med 10-100 m/bildpunkt för några få städer. På adressen `http://www.lantmateriet.se` hittar man flygbilder över hela Sverige med en upplösning om 0.25 meter (färg, storstäder) till 1 meter (landsbygd). Är man mera intresserad av planeten Mars hittar man en del svart-vita bilder med en upplösning om cirka 10 m per bildpunkt hos `http://mpfwww.jpl.nasa.gov/mgs`.

I detta datoriseringens tidevarv vill man överföra tidigare kartor i digitaliserad form för att kunna kombinera dem med färskare material. Om kurvor som t ex nivåkurvor och vägar skall representeras som polygontåg i stället för som rena punktföljder talar man om vektorisering.

I USA finns företag som ägnar sig åt att med dator färglägga gamla svart-vita spelfilmer för att tillfredsställa förmodade önskemål från en modernare publik.

Bildbehandling vinner insteg i rena konsumentprodukter. T ex har vissa videokameror programvara som skall kompensera för skakningsoskärpa.

Automatiserat seende (även kallat datorseende eller robotseende) innebär att en dator känner igen och kan skilja föremål från varandra. Aktuellt vid automatiserad avsyning och när en robot rör sig i en för människan fientlig eller oåtkomlig miljö. Det här området hamnar till stor del under begrepp som artificiell intelligens och mönsterigenkänning. Man har ännu inte nått så långt som förespråkarna hoppats.

Polisens fingeravtrycksregister hanteras med inslag av bildbehandling.

Optisk teckenigenkänning (OCR = Optical Character Recognition) har gjorts välkänd av Postverket och används praktiskt på många håll. För persondatorvärlden finns flera system som medger att en textsida skannas och omvandlas till en följd av tecken. Programvara för detta brukar numera ingå vid köp av även den enklaste skanner.

När det gäller de svenska högskolorna och universiteten har framför allt Uppsala, Linköping och Stockholm verksamhet inom bildbehandlingsområdet.

3 Utrustning för registrering av bilder

Det traditionella sättet att registrera bilder har varit fotografisk film eller på senare år videoband. Det första sättet ger som vi alla vet en oöverträffad kvalitet (upplösningen är mycket hög, tusentals punkter per cm).

För att bildbehandling i dator skall kunna ske måste bilden digitaliseras, dvs delas upp i ett antal bildpunkter. I signalbehandlingssammanhang kallas det för sampling (sample = ta ett prov). Utrustning som digitaliserar en videosignal har funnits ett bra tag och behöver inte kosta så mycket. Numera arbetar i själva verket de flesta videokameror med ett digitalt chip för bildavläsningen, men lagringen har hittills varit analog. Samma typ av krets ingår i de digitala stillbildskameror som funnits på marknaden under flera år, men först under 1998 kommit ned i sådant pris att de kan intressera vanliga konsumenter. För digitalisering av foton och andra bilder finns skannrar, som arbetar ungefär som en kopiator. Bilder som sänds från rymdfarkoster är redan när de anländer digitaliserade.

4 Programvara för bildbehandling

I detta papper utnyttjar vi framför allt MATLAB (version 6) och *xv*. Det förra har en separat verktygslåda för bildbehandling. Det senare är mera ett bildvisningsprogram, men kan litet bildbehandling. Dessutom nämner vi *xpaint* och *ppmplus*.

För PC och Mac finns rätt kraftfulla bildbehandlingsprogram antingen som "free-ware" eller "shareware". *PhotoShop* och *PaintShop* är förmodligen de mest kända för PC. *GIMP* är en ny kraftfull Gnu-produkt för UNIX-världen (även PC). Min favorit på Macsidan är *Image* gjort av en amerikansk medicinare, som användes i tidigare upplagor av denna skrift. Java har "naturligtvis" en bildbehandlingsmöjligheter, dels i grundsystemet dels i en separat Java Advanced Imaging (JAI).

4.1 xv

Står det inget om här.

4.2 MATLAB

MATLAB 6 startas med kommandot `matlab`, som ger ett användarsnitt skrivet i Java, eller med kommandot `matlab -nojvm`, som ger ett traditionellt användarsnitt. I standardverktygslådan finns några få hjälpmedel för bildbehandling. I den kommersiella extra bildbehandlingsverktygslådan finns mycket mer, oftast i form av s k m-filer, dvs man kan titta på källkoden (med `type kommandonamnet`). Prova gärna de trevliga demonstrationerna *imadjdemo*, *edgedemo* och *dctdemo* direkt eller via *demo* (och sedan **Toolboxes/Image Processing**). Mer om dessa senare.

Med kommandot `help images` får man en förteckning över de olika operationerna. Sedan kan man begära hjälp om enskilda (hjälpen är av ojämn kvalitet). De flesta funktionerna har i sedvanlig MATLAB-stil flera betydelser. Vi kan naturligtvis inte gå in på allt här utan får försöka koncentrera oss på några bitar.

Som vanligt i MATLAB representeras allt i form av matriser/vektorer. En bild är alltså en matris, en färgtabell en vektor, etc. MATLAB kan hanterar svart-vita (binära bilder), RGB-bilder, gråskalebilder och indexerade bilder (bilder baserade på färgtabell). I det arbetet finns en del automatik, som kan skapa förvirring (t ex omvandling mellan olika format). En speciell egenhet, som också kan förvirra, är att MATLAB sällan visar bilden i den rätta storleken, utan förstorar/förminskar den. Man kan dock med kommandot `truesize` få rätt storlek.

Inläsning av bild till MATLAB:

```
>> [BILD,CMAP]=imread('bildfilen','format');
```

Bilden hamnar i matrisen BILD (som får rätt storlek) och färgtabellen i vektorn CMAP (för RGB-bilder räcker det att skriva BILD = ...). BILD innehåller heltalsvärden mellan 0 och 255 (eller lägre; eller i vissa fall reella tal mellan 0 och 1). Färgtabellen består av ett antal rader med tre tal mellan 0 och 1, som vardera anger intensiteten (1 är högst) för rött, grönt respektive blått. Vi kan naturligtvis skriva ut innehållet i BILD och CMAP i terminalfönstret, men det blir väldigt många tal (om inte bilden är liten). MATLAB stöder olika varianter av formaten BMP, JPEG, PCX, TIFF, XWD och HDF. Däremot stöds från och med version 5.0 inte längre GIF.

Olika sorters bilder i MATLAB (B bildvariabeln):
Färgtabellbilder: Matriselementen är heltal mellan 0 och 255. Ritas med `imshow(B,CM)`. Alternativt kan man använda kommandon från grund-MATLAB: `colormap(CM); image(B)`, där det första kommandot gör CM till aktuell färgtabell. Om det utelämnas används gällande färgtabell.

Svart-vita (binära) bilder: Elementen är antingen 0 (svart) eller 1 (vitt). Ritas med `imshow(B)`.
Gråskalebilder: Elementen är tal mellan 0.0 (svart) och 1.0 (vitt). Ritas med `imshow(B,N)`, där N anger önskat antal gråskalenivåer. Om N utelämnas används N=256.
RGB-bilder: Dessa beskrivs med en MxNx3-matris BILD, där BILD(:,:,1) är intensitetsmatrisen för R-komponenten (0-255), BILD(:,:,2) och BILD(:,:,3) motsvarande för G- och B-komponenterna och ritas med `imshow(BILD)` eller `image(BILD)`.
Även helt godtyckliga matriser kan ritas upp som bilder. Värdena skalas därvid till ett indexintervall och befintlig färgtabell används.

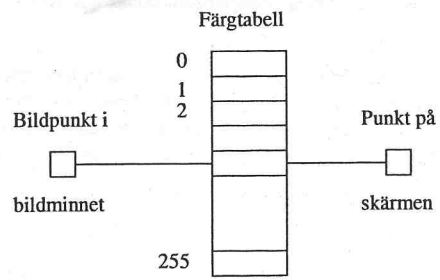
Det finns ett flertal konverteringsfunktioner, t ex `I=im2bw(BILD,CM,1.0)` som omvandlar en färgtabellbild till en svart-vit.

Exempel: `B=[1,2;3,4]; C=[0,1;1,0]; D=[0,0.75,0.90,0.95];` ger alla bilder.
Likaså `B=round(255*rand(200,200)); B=rand(200,200);`
`B=round(rand(200,200));`

5 Bilder

Vi skall här ta upp operationer som ändrar ljushet och kontrast i en bild, tar bort brus, upptäcker kanter med mera. Våra operationer är lokala till sin karaktär, dvs vid övergången
ursprunglig bild -> ny bild
beror en bildpunkt i den nya bilden bara av några få punkter i den gamla.

Färgtabell
Numera arbetar de flesta datorskärmar med RGB, dvs 24 bitar/bildpunkt. Tidigare användes färgtabell och typiskt gick det åt 8 bitar/bildpunkt. I figuren illustreras hur en färgtabell fungerar:



Vi kan alltså genom att ändra i enda position i färgtabellen ändra många punkter på skärmen. Närmare bestämt alla de punkter som har samma index (däremot inte säkert alla punkter med en viss färg eftersom olika index kan motsvara samma färg; bör dock ej eftersträvas). Utan färgtabell är detta mycket omständligare. Man måste gå igenom samtliga punkter i bilden, vilket tar längre tid. En nackdel med färgtabellen är dock att lokala förändringar inte är möjliga. I gråskalesituationen tänker man sig ofta att färgtabellen är inställd så att 0->255 motsvarar svart->vitt.

På liknande sätt kan en bild lagras i (oftast) bantat format med färgtabell. Man brukar säga att bilden är indexerad.

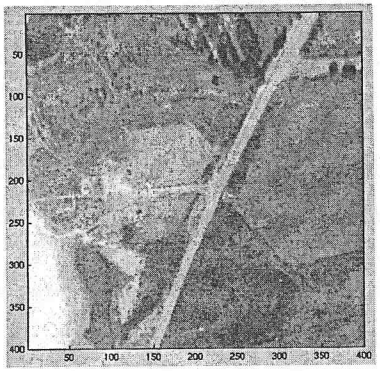
Avläsning av bildpunkter

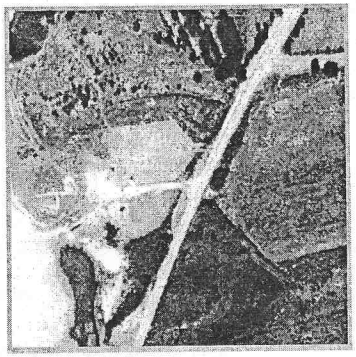
I OpenGL (som man kanske inte tar till för bildbehandling) kan vi med rasterkopieringsfunktioner som `glReadPixels` avläsa färgen i en bildpunkt. I MATLAB kan man använda någon version av `impixel`.

Histogram

Ett histogram är ett stapeldiagram som i det här fallet visar hur frekventa de olika gråvåerna (eller färgerna) är. Stapelhöjden anger andelen eller antalet punkter med en viss gråhet. Ett sådant diagram bildar ofta utgångspunkten för vidare arbete. För bilden nedan till vänster visas histogrammet längst upp till vänster på nästa sida. Eftersom bilden inte innehåller några riktigt svarta punkter, finns inga staplar i intervallet [0, 0.15].

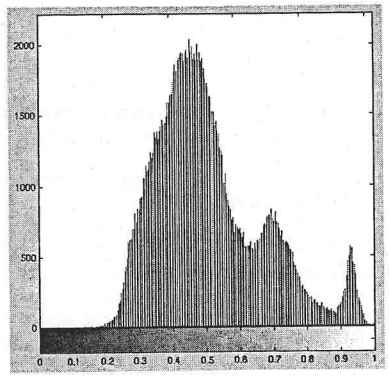
Histogramutjämning är en teknik som siktar på att varje del av gråskalan skall få ungefär lika många punkter i bilden och upplevs ofta som en bildförbättring. Nedan ser vi till vänster en gråskalebild (`BAY_1.bmp`) och till höger motsvarande bild efter histogramutjämning. I detta fall är utgångsbilden av så hög kvalitet att histogramutjämningen inte lyckas. Ändringen kan genomföras i färgtabellen (*xv*) eller i de lagrade värdena (MATLAB).

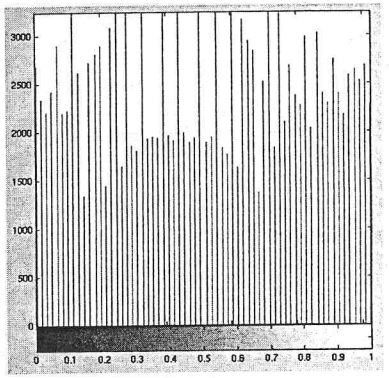




På nästa sida visas histogrammen för de båda bilderna. Följande MATLAB-kommandon utförda successivt visar bilderna och histogrammen (`imhist(bild,N)` fördelar bilden på N nivåer, utelämnas N så används värdet 64).

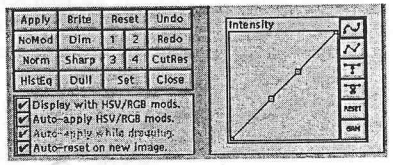
```
>> [BILD,CM]=imread('BAY_1.bmp','bmp'); % Inläsning av bilden
>> imfinfo('BAY_1.bmp')                 % Ger information om bilden(filen)
    ...
    ColorType: 'indexed'                % Säger att bilden har färg-
    NumColormapEntries: 230             % tabell med 230 ingångar
    ...
>> G=ind2gray(BILD,CM);                 % G blir gråskalebild, går utan
>> imhist(G);                           % Ritar histogrammet för G
>> GMOD=histeq(G);                      % GMOD är G histogramutjämnad
>> imhist(GMOD);                        % Ritar histogrammet för GMOD
>> imshow(GMOD);                        % Rita GMOD (gråskalebild)
```



Ljushets- och kontrastförändring

Förhållandet mellan indexvärde i en bildpunkt och punktens intensitet bestäms av färgtabellen. Initialt är den förmodligen som i högra delen av figuren nedan.



Bilden är hämtad från *xv*. Ett enkelt sätt att invertera bilden är att dra kurvan så att den följer den andra diagonalen. Knapparna **Brite** och **Dim** ökar respektive minskar intensiteten i bilden (kurvan flyttar sig åt vänster resp höger). Knapparna **Sharp** och **Dull** ökar respektive minskar kontrasten i bilden (kurvans lutning ökar resp minskar).

Falska färger

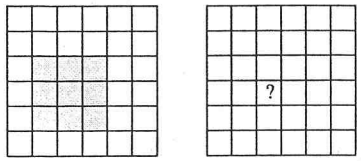
Ibland vill man ge ökat uppmärksamhetsvärde åt någon del av en gråskalebild. Om delen har en enhetlig gråhet, så kan det göras med färgtabellen.

Tröskling (eng. threshholding)

Tröskling innebär att en gråskalebild överförs till en bild med bara svart och vitt. Det går till så att man väljer ett index (0-255) eller ett intensitetsvärde. I båda fallen talar vi om ett **tröskelvärde**. Alla värden mindre än tröskelvärdet får motsvara svart och alla övriga vitt. Tillfälligt kan ändringen göras med färgtabellen, men för permanent effekt måste bilden gås igenom punkt för punkt.
För lyckat resultat måste bilden vara lämplig. Valet av tröskelvärde sker lämpligen utifrån histogrammet.

Brusreducering

Ett antal bearbetningar kan utföras med operationer av följande typ. Vi tänker oss en kärna i form av en matris av dimension mxm med heltal. Denna placeras på originalbilden (se figur på nästa sida) och får flyta över den när vi beräknar den nya bilden. Säg att vi vill beräkna den nya bildens värde (intensitet; i fallet färgtabell antas den vara linjärt inställd) i den ?-märkta punkten med koordinater (x,y). Kärnan placeras då med sin mittpunkt över motsvarande punkt i originalbilden (ett skäl till att talet m rimligen är udda; i figuren är det 3).



Det nya värdet fås nu som

$$\sum_{i,j} c[i,j]bild[x+i,y+j]$$

där c[i,j] står för talen i kärnan.

Vid brusreducering (utjämning, medelvärdesbildning) bildar man ett medelvärde av punkterna i närheten av en punkt. Kärnan kan t ex vara som i vänstra figuren nedan. Talen i matrisen skall divideras med talet nedanför, dvs 8, så att värdet ligger i samma intervall som förut. Resultat utanför de

	1	
1	4	1
	1	

8
Utjämnande

	1	
1	-4	1
	1	

4
Laplace

1	2	1
0	0	0
-1	-2	-1

4
Kant (y-led)

-1	0	1
-2	0	2
-1	0	1

4
Dito (x-led)

tillåtna bildvärdena ersätts med minsta resp största värde.

Man kallar ofta operationer av denna typ för filter. Det utjämnande filtret räknas som ett lågpass-filter, eftersom störningar (högre frekvenser) dämpas, medan långsamma variationer släpps igenom (kvarstår).

Exempel: En bild B[1:N,1:N] kan om vi bortser från kanterna utjämnas med

```
B[2:N-1,2:N-1] = (4*B[2:N-1,2:N-1]+B[1:N-2,2:N-1] + B[3:N,2:N-1] +
                  B[2:N-1,1:N-2] + B[2:N-1,3:N])/8;
```

För att de aritmetiska operationerna skall fungera måste den vara av reell typ. Om bilden är av heltalstyp kan omvandling ske med `B=im2double(B);`.

Kantdetektering

Laplaceoperatorn (andra från vänster i figuren ovan) är i stället ett högpassfilter. Långsamma variationer dämpas nämligen. Talen i matrisen motiveras av att den skall approximera

$$\frac{\delta^2}{\delta x^2} + \frac{\delta^2}{\delta y^2}$$

Kanske känner du igen koefficienterna från någon kurs i numerisk analys. Laplaceoperatorn detekterar kanter.

Två andra kantdetektorer finns till höger i figuren. Den första detekterar horisontella kanter och den andra vertikala. Vill man samtidigt detektera båda slagen, kan man som nytt bildvärde ta maximum av absolutvärdena som respektive operator ger.

I en rent svart-vit bild är det principiellt lätt att detektera kanter. Vertikala och sneda kanter kan lätt identifieras med en skanning-process linje för linje. Denna process missar däremot horisontella kanter.

Andra operationer

En mängd andra och skickligare operationer av detta slag finns. Se t ex MATLABs kommandoförråd eller menyerna i *xv* och *xpaint*.

Halvtoning och dittring

Kursböckerna (Hearn/Baker eller Hill) berör litet grand problemet att återge en gråskalebild på utrustning med t ex enbart svart och vitt (vi har eller kommer att diskutera det på en föreläsning). Allmännare gäller det att återge en färgbild på en utrustning med begränsat antal färger, t ex en 24-bitars bild på 8-bitars utrustning. Operationerna halvtoning och dittring kan vanligen ses som linjära lokala operationer. Diffusionsmetoden på sid 521 i HB är dock olokal.

Expansion och krympning (tunning)

Antag att vi har en binärbild med ett objekt där det finns ett eller flera hål på grund av någon störning. Ett sätt att bli av med hålen är att **expandera** objektet ett antal steg och sedan **krympa** det lika många steg. Om vi antar att objektet är svart, så innebär expansion att man svartlägger vita punkter som ligger intill svarta och krympning att man i stället vitlägger svarta punkter som ligger nära vita. Praktiskt kan krympning ske genom att man skannar bilden linje för linje. Påträffas en svart punkt så undersöker man om det finns någon vit granne (4-grannar eller ev 8-grannar). I så fall raderas motsvarande punkt i en kopia av bilden, annars får den förbli svart. Vid sådan krympning kan ett objekt brytas ned i mindre delar. Genom tilläggsvillkor kan man se till att objektet hålls ihop vid krympningen. Detta används vid krympning till en punkt eller till ett s k skelett, se den fjärde figuren nedan. Till vänster visas i en sekvens om tre bilder resultatet av en expansion med 10 steg följd av en lika lång krympning. Längst till höger syns resultatet av en kantdetektering.

I MATLAB finns operationen `dilate` för expansion och `erode` för krympning. Men båda utgår från att objektet är vitt (dvs expanderar respektive krymper vitt). För att vi skall få ovanstående (tryckteknisk) mera naturliga tolkning, inverterar jag därför färgerna före och efter operationen. Nedan visas koden för var och en av figurerna. .

```
>> [A,CM]=imread('Bokstav.bmp','bmp');    % Läser
>> imshow(A);

>> ATJOCK=~imdilate(~A,ones(8,8));        % Expanderar svart
>> imshow(ATJOCK);

>> AFIXAD=~imerode(~ATJOCK,ones(8,8));    % Krymper svart
>> imshow(AFIXAD);

>> ASKEL=~bwmorph(~A,'skel',Inf);         % Krymper till skelett
>> imshow(ASKEL);

>> AEDGE=~edge(~A);                       % Kanter
>> imshow(AEDGE);
```

6 Geometriska operationer

Den enklaste formen av geometrisk operation är förstoring. Man markerar då en del av bilden som sedan visas uppförstorad. Detta kallas ofta "zooming". Vanligen går det till så att de enskilda bildpunkterna visas uppförstorade (s k pixelförstoring), dvs väljs tillräckligt hög förstoringsgrad ser man bara några få bildpunkter i den ursprungliga bilden. Zooming åstadkommes t ex med någon variant av en rasterkopieringsfunktion (*glCopyPixels/glDrawPixels* i OpenGL). I mycket speciella system finns hårdvarustöd för zooming i videokretsen. Det gäller ju huvudsakligen att förlänga varaktigheten i signalen från en bildpunkt.

Vid denna typ av förstoring framträder bildens digitala karaktär alltmer. Bättre är att vid förstoringen använda interpolation.

Syftet med en allmännare geometrisk operation är i allmänhet att ta bort geometriska deformationer i en registrerad bild. Sådana kan t ex vållas av optiken i en vanlig kamera eller av att man fotograferar en krökt yta. Man måste naturligtvis då veta eller räkna fram hur den korrigerande transformationen skall gå till. Rymdteleskopet Hubble skickades upp med en felaktig slipad spegel. Med bildbehandling kunde man i efterhand i viss mån kompensera för det. Resultaten blev ännu bättre när spegeln sedermera byttes ut.

Vid en geometrisk transformation har vi en funktion XD = F(X) som avbildar punkter X i utgångsbilden på punkter XD i den transformerade bilden. Det är ofta lämpligt att tänka sig reella koordinater. Vi låter funktionen arbeta på något område, i enklaste fallet en rektangel. Ett exempel är F(X) = AX + B, där A är en matris och B en vektor. Denna linjära transformation avbildar en rektangel på en rektangel. I detta exempel ryms förstoringar och förminskningar. Ett annat exempel är F(X) = (x*x,y*y). Ibland är det behändigt att i stället se x och y som komponenterna i ett komplext tal z och använda funktioner F(z) som ger nya komplexa tal. Ett exempel är F(z) = z*z = (x+iy)*(x+iy)= x*x-y*y+2ixy.

Vid uppritandet av den transformerade bilden kan man gå tillväga på några olika sätt.

- För varje bildpunkt X i källan beräkna motsvarande XD i destinationen och markera den med samma gråvärde som X har. Detta sätt är inte lyckat eftersom det vid uppförstoring kommer att finnas ett antal bildpunkter som inte markeras (glapp i den nya bilden).
- Motsvarande men välj punkterna X tätare. Om X inte är en bildpunkt
  a) välj gråvärde som i den närmsta bildpunkten (sämst)
  b) välj gråvärde med bilinjär eller bikubisk interpolation (bättre)
  Markera den bildpunkt i destinationen som ligger närmst XD.
  Svårigheten med denna teknik är att välja rätt täthet.
- För varje bildpunkt XD i destinationen beräkna motsvarande punkt X (ej nödvändigtvis bildpunkt) med $X = F^{-1}(XD)$, dvs gör den inversa transformationen. Tag reda på gråheten i X som i punkt 2 och markera XD. Detta sätt ger ett gott resultat men kräver ofta en större arbetsinsats.

7 Extraktion av egenskaper

I en bild kan finnas ett antal objekt som vi har något intresse av. Det kan t ex röra sig om celler i en mikroskopbild av ett preparat eller skogshyggen i en satellitbild. Vi kanske vill ha reda på antalet objekt och de enskilda objektens area. Detta kan givetvis ske med manuella insatser (och någon sådan blir i allmänhet alltid nödvändig), men vi eftersträvar så hög automatiseringsgrad som möjligt.

Låt oss som vanligt förenkla och anta att det rör sig om en binärbild och att objekten är svarta (i enklaste fall kan detta ske med tröskelsättning och expansion/krympning för att få bort hål i objekten på grund av störningar). Vi antar också att objekten är sammanhängande på ett sådant sätt att den från datorgrafiken välkända enkla fyllnadsproceduren kan användas.

Räkning av antal objekt
Följande program beräknar med en mycket enkel metod antalet objekt. Bilden gås igenom punkt för punkt och rad för rad. När en svart bildpunkt påträffas, raderas med fyllnadsproceduren *Fill* alla punkter i det objekt inom vilket punkten ligger. Vi kommer därför aldrig att stöta på någon ytterligare punkt i det objektet och kan således samtidigt räkna upp en objekträknare *count*. Vi förutsätter att objekten ligger innanför en rektangel *r*.

```
void Fill (int x,int y)
{
   if (GetPixel(x, y)) {
       MoveTo(x,y); LineTo(x,y);
       Fill(x + 1, y);Fill(x - 1, y);
       Fill(x, y + 1); Fill(x, y - 1);
    }
}

void main()
{
  /* Deklarationer*/
  count = 0;
  for (y = r.top; y <=r.bottom; y++) {
       for (x = r.left; x <= r.right; x++) {
           if (GetPixel(x, y)) {
              /*Objekt hittat; radera alla punkter i objektet */
              count = count + 1;
              PenPat(White); // Rita med vitt
              printf("Found object %d\n", count);
              Fill(x, y);
              PenPat(Black);  // Rita med svart
            }
        }
   }
   printf("Antal objekt= %d \n", count);
}
```

Omkrets och area
Samtidigt som objekten räknas kan vi registrera en punkt per objekt och senare med någon av de i datorgrafikdelen nämnda algoritmerna beräkna t ex omkrets och area. Dessa beräkningar kan naturligtvis också göras samtidigt med räknandet av antalet. MATLAB har funktioner för bl a areor av objekt i svart-vita bilder.

När det gäller igenkänning av objekt, t ex avgöra om ett objekt är bokstaven K eller A, så kan man i mycket enkla fall utgå från någon sorts formfaktor. Men i allmänhet krävs avancerade tekniker.

8 Bildkomprimering

Bilder kräver betydande minnesutrymme. Därför försöker man på olika sätt att komprimera dem dels vid lagring, dels vid överföring. Man använder dels standardprogramvara för komprimering av filer, dels specialprogramvara. Typiskt uppnår man komprimeringsfaktorer på mellan 1:2 och 1:10. Faktorn beror dock mycket på bildens utseende. En fullständigt slumpmässig bild blir gärna större

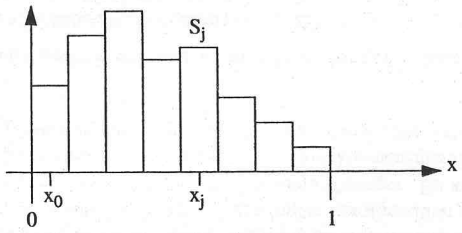
efter komprimering. Om man har flera likartade bilder efter varandra (film, bildtelefoni), kan man dock få ännu högre faktorer genom att utnyttja koherensen mellan bilderna. För stillbilder är JPEG ett vanligt komprimeringsformat (förstörande). För rörliga bilder i stället MPEG.

Exempel: En svart-vit 512*342-bild innehåller 24 kB, medan en svart-vit 1024*1024-bild innehåller 125 kB och en lika stor färgbild med 256 färger kräver 1000 kB och med 24-bitar 3000 kB. En röntgenbild med fotografisk kvalitet skulle kräva åtskilliga MB, vilket är en förklaring till att man i sjukvården alltjämt har fotoarkiv.

9 Fourieranalys

Fourieranalys handlar om att dela upp en signal (funktion) i toner. Ofta handlar det om kontinuerliga signaler, men för vår del är det diskreta som gäller. Uppdelningen kan göras på många olika sätt. Vi tar här bara upp DCT (Diskret CosinusTransform) i den form som MATLAB använder. Vi siktar mot DCT för bilder, dvs för funktioner i två variabler, men startar med endimensionella fallet.

Vi har en signal samplad i N ekvidistanta punkter numrerade 0, 1, 2, ..., N-1. Det är ofta underförstått nedan att N är en 2-potens, dvs $N=2^p$, där p är ett heltal>0. Vi lägger på ett virtuellt koordinatsystem så att punkterna motsvarar $x_j = (j+1/2)/N$, j = 0,1,...,N-1. Vi skriver S_j eller $S(x_j)$ för signalvärdena.



Vi vill nu bestämma de s k fourierkoefficienterna F_j, j = 0,1, ..., N-1, så att

$$S_j = \sum_{k=0}^{N-1} F_k \cos(k\pi x_j)$$

Genom att multiplicera båda leden med $\cos(m\pi x_j)$ och summera över j, finner man med litet möda att

$$F_m = C_m \sum_{j=0}^{N-1} S_j \cos(m\pi x_j)$$

där $C_m = 1/N$ för m=0 och $C_m = 2/N$ för 0<m<=N-1. Speciellt är F_0 medelvärdet av signalvärdena.

Vi visar nu ett exempel i MATLAB för N=4. Att det ibland står j+1 etc som index beror på att MATLAB indexerar från 1.

```
>> S=[8,8,8,6];
>> N=4;
>> X=1.0/(2*N):1.0/N:0.99
X =    0.1250    0.3750    0.6250    0.8750
```



```
>> for m=0:3
       F(m+1)=2*sum(S .* cos(m*pi*X))/N;
   end

>> F(1)=F(1)/2      % Korrigerar p g a att C0 = 0.5*C1
F =    7.5000    0.9239   -0.7071    0.3827
>> K=0:N-1
K =     0     1     2     3
>> for j=0:N-1
       INVERS(j+1)=sum(F .* cos(K*pi*X(j+1)));
   end
>> INVERS
INVERS =    8.0000    8.0000    8.0000    6.0000
```

Sedan gör vi samma sak med MATLABs inbyggda DCT-funktioner. Av olika skäl tar man inte med faktorn C_m här, dvs värdena har multiplicerats med 2N resp N i förhållande till tidigare.

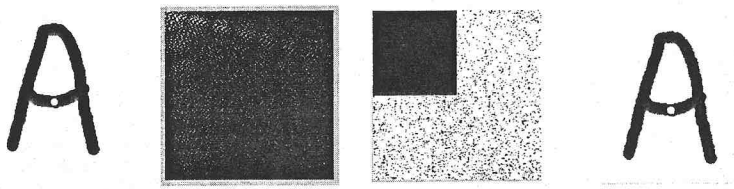
```
>> F=dct(S)
F=   60.0000    3.6955   -2.8284    1.5307
>> INVERS=idct(F)
INVERS =     8     8     8     6
```

För bilder är signalen i stället en funktion av två variabler. Analogt betecknar vi de samplade värdena med S_{ij} och fourierkoefficenterna med F_{km}. Om vi förenkelhets skull antar att bilden är kvadratisk ser det ut så här:

$$S_{ij} = \sum_{k=0}^{N-1} \sum_{m=0}^{N-1} F_{km} \cos(k\pi x_i) \cos(m\pi y_j)$$

Och man kan ange ett uttryck för fourierkoefficienterna ungefär som förut (vi hoppar över det). MATLAB har en färdig funktion *dct2* som ger dem och en *idct2* som transformerar tillbaka.

Ett MATLAB-exempel till. Till vänster visas en BMP-bild med storleken 200x200. Tillverkad med *xpaint*. Den lästes in i MATLAB.



```
>> [G,CM]=imread('Bokstav.bmp');
>> CM                              % Visar att färgtabellen har två ingångar
CM =                               % en för svart och en för vitt
     0     0     0
     1     1     1
>> imshow(G,CM);                    % Vänstra bilden
>> GF=dct2(G);>> IGF=idct2(GF);    % G och IGF är identiska
>> imshow(GF)                      % Andra bilden, fourierkoefficienterna
```

Nu modifierar vi fouriermatrisen kraftigt.

```
>> GFM=GF; GFM(1:100,101:200)=zeros(100); % 100x100 matris med nollor
>> GFM(101:200,1:100)=zeros(100); GFM(101:200,101:200)=zeros(100);
>> imshow(100*abs(GF-GFM))    % Vi visar skillnaden förstorad, 3:e bilden
>> IGFM=idct2(GFM);
>> imshow(IGFM)               % Högra bilden
```

10 Några av MATLABs bildbehandlingsdemo

På följande sidor följer några skärmbilder med bildbehandlingdemonstrationer från MATLAB.

1. `imadjdemo` med knappvalet **Intensity Adjustment**. Efter ett stegs kontrastökning, dvs **+ Contrast** en gång. Från att ha varit diagonal roterar den runt mittpunkten motsol, dvs antalet svarta bildpunkter och antalet vita ökar. Relativt sett minskar därmed alla andra. + Brightness hade i stället ökat intensiteten för alla punkter, vilket gör att transformationskurvan höjs och staplarna i histogrammet åker åt höger.

2. `imadjdemo` med knappvalet **Histogram Equalization**.

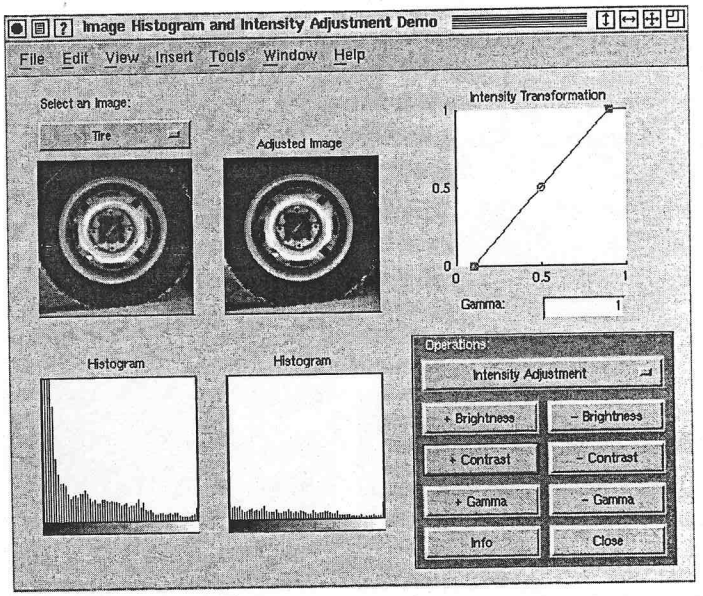
3. `nrfiltdemo`. Originalbilden störs med någon form av brus (i exemplet Salt & Peppar, som innebär full störning eller ingen alls; ungefär en femtedel av bildpunkterna störs). Därefter försöker man med något filter reducera bruset.

4. `dctdemo` visar även en del av funktionssättet i vid JPEG-kompression. Vi har en 64x64 bild av en ros. Den delas in i 64 stycken block om 8x8 bildpunkter. Vart och ett av dessa block kan representeras med 64 fourierkoefficienter (diskret cosinustransform). Om vi använder alla dessa vid återtransformationen återfår vi naturligtvis den ursprungliga bilden. I figuren har man med hjälp av reglaget valt att bara använda de fyra "viktigaste", vilket gör att resultatet vid återtransformationen blir litet fel. Felet visas också i form av en bild.

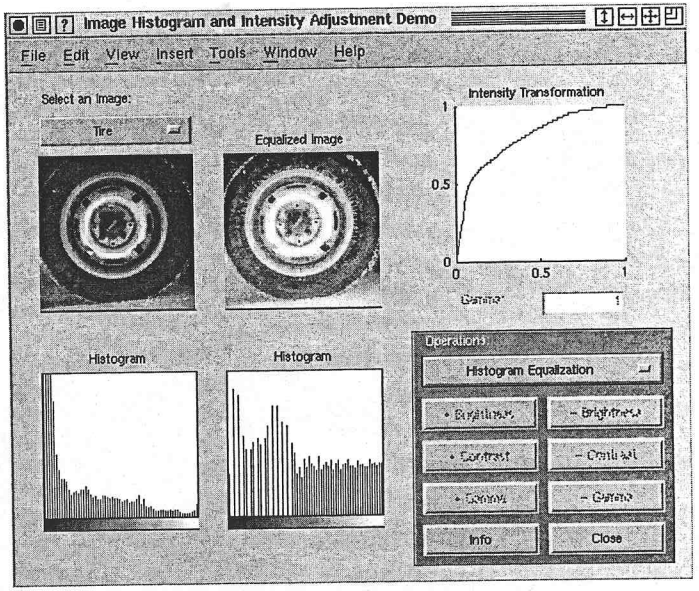
5. `landsatdemo`: This demo allows you to experiment with creating color composites from Landsat Thematic Mapper data. Landsat data consists of 7 spectral bands that each reveal different features of the region that is imaged. The data is read into a 512-by-512-by-7 array. To create a color composite, we form an RGB image by assigning spectral bands to red, green, and blue intensities.
Try out some common color composites by clicking on the radio buttons. The numbers in square brackets map the spectral bands to red, green, and blue. The array [3 2 1] means band 3 will be shown as red intensities, band 2 will be shown as green intensities, and band 1 will be shown as blue intensities.
"True Color [3 2 1]" - shows what our eyes would see from an airplane.
"Near Infrared [4 3 2]" - shows vegetation as red, water as dark.
"Shortwave Infrared [7 4 3]" - shows changes due to moisture.

6. `edgedemo` visar kantdetektering med olika metoder.

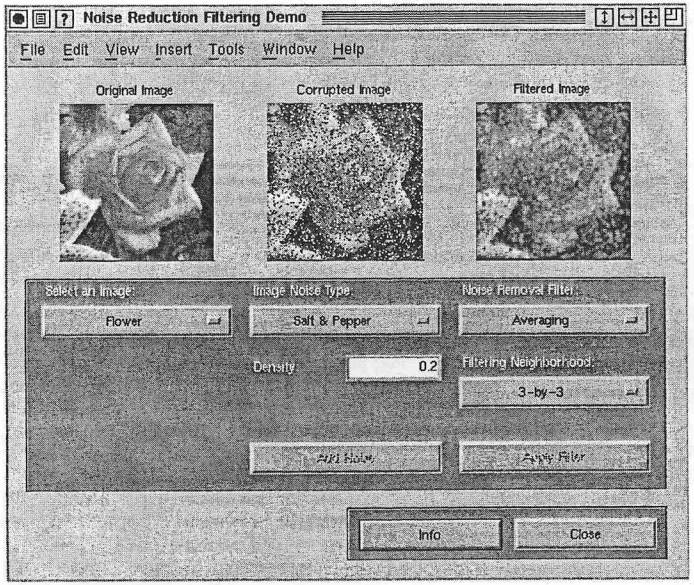
1



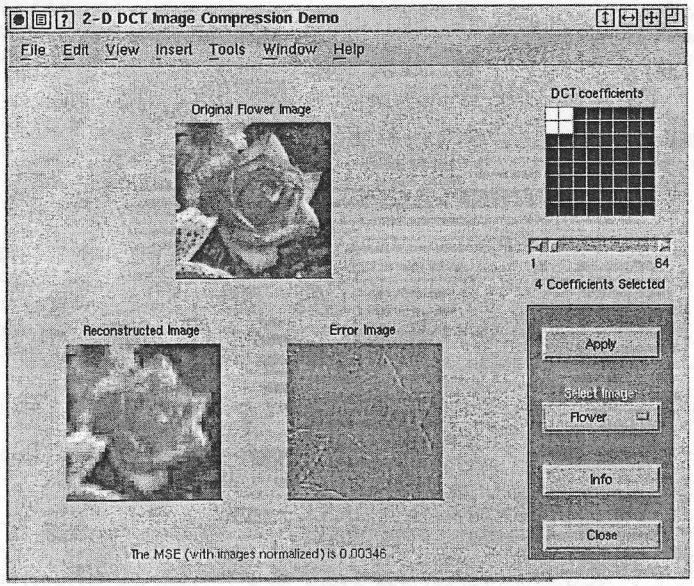
2



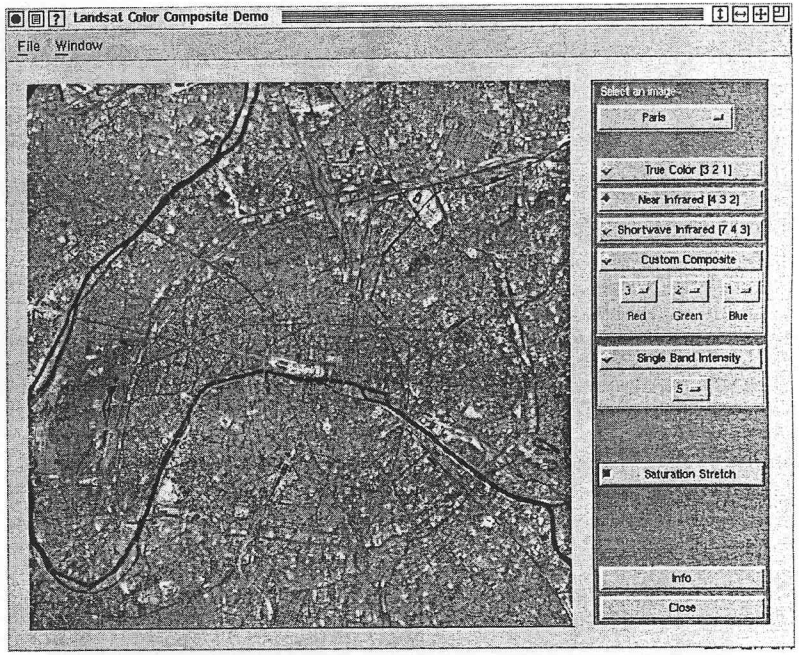
3


Noise Reduction Filtering Demo
File Edit View Insert Tools Window Help
Original Image
Corrupted Image
Filtered Image
Select an Image:
Flower
Image Noise Type:
Salt & Pepper
Noise Removal Filter:
Averaging
Density:
0.2
Filtering Neighborhood:
3-by-3
Add Noise
Apply Filter
Info
Close

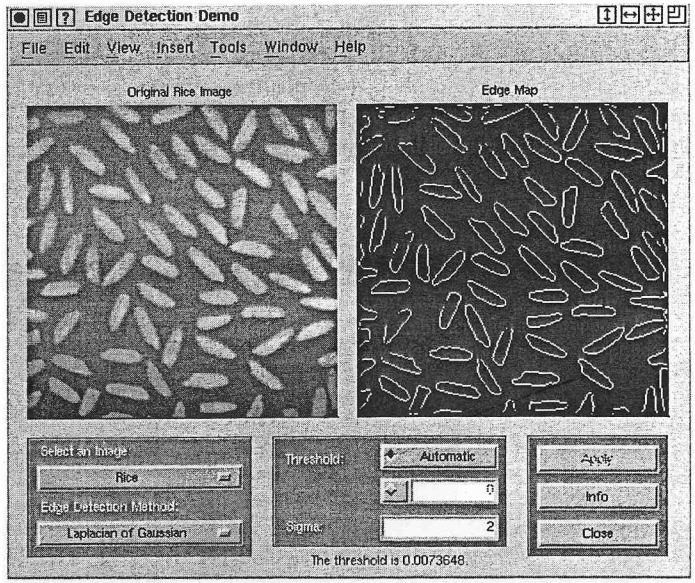
4


2-D DCT Image Compression Demo
File Edit View Insert Tools Window Help
Original Flower Image
DCT coefficients
1
64
4 Coefficients Selected
Reconstructed Image
Error Image
Apply
Select Image:
Flower
Info
Close
The MSE (with images normalized) is 0.00346

5


Landsat Color Composite Demo
File Window
Select an Image:
Paris
True Color [3 2 1]
Near Infrared [4 3 2]
Shortwave Infrared [7 4 3]
Custom Composite
3
2
1
Red
Green
Blue
Single Band Intensity
5
Saturation Stretch
Info
Close

6


Edge Detection Demo
File Edit View Insert Tools Window Help
Original Rice Image
Edge Map
Select an Image:
Rice
Edge Detection Method:
Laplacian of Gaussian
Threshold:
Automatic
Sigma:
2
Apply
Info
Close
The threshold is 0.0073648.