Digital Billed Manipulerings Software (DIMS) Projekt 2, 02633, januar '10

Projektet laves i grupper på 2-3 personer.
Rapporten skal afleveres med angivet gruppenummer
senest fredag den 15. januar, kl. 12:00 til sekretær Mette Larsen, rum 011, bygning 321.
Rapporten for Projekt 1 skal også returneres senest dette tidspunkt

Mål

Målet med dette projekt er at udvikle et MATLAB program-pakke, som kan benyttes til at lave otte forskellige operationer på et 8-bit gråtone- eller et 24-bit farvebillede og udlæse resultaterne i form af et billed-galleri. I projektet skal man lave et underprogram for hver af de otte operationer, samt udvikle og implementere en algoritme for en del af hovedprogrammet (den såkaldte 'driver'), herunder en passende datastruktur for billed-galleriet.

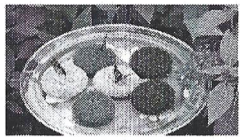
Kravspecifikation

- Vi vil arbejde med gråtone-billeder, som MATLAB -funktionen `imread` læser ind i to-dimensionale (2D) arrays af typen `uint8`, samt 'truecolor' (dvs. 24 bit) RGB-farvebilleder, som indlæses i 3D arrays af typen `uint8`. For begge typer af arrays gælder der, at det i MATLAB er tilladt at referere til elementer via tre indices, hvis blot det tredje indeks er 1 eller ':' i tilfælde af 2D arrays, og dette skal udnyttes.
- Program-pakken skal bestå af otte **funktioner** som hver kan udføre en billed-operation, og et 'driver' **script** som tager sig af interface't til brugeren.
- De otte operationer pakken skal tilbyde er:
  1. *grayscale*: danner et gråtone-billede svarende til et givet billede.
  2. *negative*: danner negativet af et givet billede.
  3. *warp*: tilføjer en fiskeøje-effekt med specificeret niveau til et givet billede.
  4. *frame*: tilføjer en ramme af specificeret bredde og farve til et givet billede.
  5. *mirror*: erstatter højre/venstre halvdel af et givet billede med spejlbilledet af den anden halvdel.
  6. *jail*: tilføjer lodrette bjælker med specificeret afstand til et givet billede.
  7. *superpixel*: tilføjer en "pixelerings-effekt" til et givet billede.
  8. *chess*: tilføjer en skakbræt-effekt til et givet billede.
- Derudover skal pakken acceptere følgende tre kommandoer:
  1. *newX*: udskifter det billede X, som operationerne skal foretages på, med et billed lagret i galleriet eller på en fil.
  2. *showGal*: fremviser billed-galleriet via MATLAB funktionen `subplot`.
  3. *stop*: standser programudførelsen.

Figuren på næste side viser et galleri, som skal kunne laves ved hjælp af programpakken. Indholdet af et figur-vindue er blevet printet via MATLAB -kommandoerne `orient landscape` og `print -djpeg` *filnavn*.

Original Image

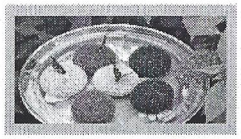
grayscale



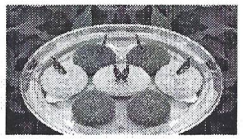
negative

warp, level=2

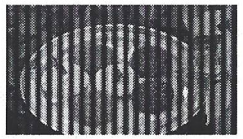
frame,width=40,col=[1 250 200]



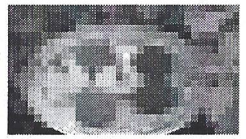
mirror, left-to-right



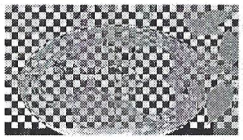
jail, distance=40



superpixel, size=25



chess, size=25



newX (from file)

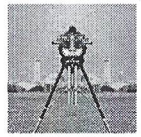
grayscale

negative

warp, level=2

frame,width=40,col=[100]

mirror, right-to-left

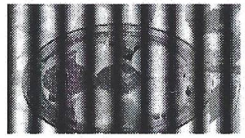


jail, distance=40

superpixel, size=25

chess, size=40

jail on newX (from gallery)



Specifikation af funktionerne for de otte operationer

I det følgende betegner `dimX` et 1x3 array indeholdende højden, bredden og dybden af et array `X`, der indeholder et billede.
Hvis det er et gråtone-billede, er `dimX(3)` lig med 1, ellers er `dimX(3)` lig med 3.

1. **grayscale**-operationen.

Funktions-navn:	`grayscale`
Input-parametre:	`X` er et 2D eller 3D array af typen `uint8` med et billede. `d` er værdien af `dimX(3)`.
Output-parameter:	`Y` er et 2D array af typen `uint8` med et billede der er gråtone-udgaven af `X`.
Formål:	At beregne gråtone-billedet svarende til `X`.
Filer udleveret:	`grayscale.m` *indeholder hele funktionen.*
Begrænsninger:	*Der må ikke ændres noget ved funktionen.*

   Algoritme:
   Er implementeret i filen `grayscale.m`
   Der anvendes samme beregning som den MATLAB -funktionen `rgb2gray` (der findes i de nyere versioner af MATLAB) benytter til at konvertere et rgb-farvebillede til et gråtone-billede.

2. **negative**-operationen.

Funktions-navn:	`negative`
Input-parameter:	`X` er et 2D eller 3D array af typen `uint8` med et billede.
Output-parameter:	`Y` er et array af samme dimensioner og type som `X` med et billede der er negativet til `X`.
Formål:	At beregne negativet til billedet `X`.
Filer udleveret:	`negative.m` *indeholder funktionens hoved.*
Begrænsninger:	*Der må ikke bruges løkker (loops) i funktionens krop.*

   Algoritme:
   Givet et billed-array `X`, så er det (i, j, k)-te element af det negative billede `255 - X(`i, j, k`)`.

   Hovedet for denne funktion er givet i filen `negative.m`. Men der mangler nogle kommentarer samt funktionens krop. Man skal indsætte de manglende dele. De steder, hvor dette skal gøres, er tydeligt markeret i filen.

3. **warp**-operationen.

Funktions-navn:	`warp`
Input-parametre:	Man skal selv specificere de nødvendige parametre.
Output-parameter:	`Y` er et 2D eller 3D array af typen `uint8` med et billede svarende til `X` med tilføjet warp (eller fiskeøje) effekt.
Formål:	At tilføje warp effekt til billedet i `X`.
Filer udleveret:	`warp.m` *indeholder funktionens krop.*
Begrænsninger:	*Der må ikke ændres noget i funktionens krop.* `help warp` *skal give nyttig information (jf. s.7 nederst).*

   Algoritme:
   Er implementeret i filen `warp.m`.

   Bemærk at i den udleverede kode skal variablen `p` indeholde et reelt tal som er positivt, nul eller negativt, og som kontrollerer graden af effekten. Værdien af `p` skal specificeres af brugeren.

   Funktionens krop er givet i filen `warp.m`. Men hovedet samt kommentarer mangler. Man skal indsætte de manglende dele. De steder, hvor dette skal gøres, er tydeligt markeret i filen.

4. **frame**-operationen.

Funktions-navn:	`frame`
Input-parametre:	`X` er et 2D eller 3D array af typen `uint8` med et billede. `dimX` er et 1x3 array indeholdende dimensionerne af `X`. `width` er et bredden af rammen, målt i antal pixels. `colors` er et `1xdimX(3)` array specificerende rammens farve.
Output-parameter:	`Y` er et array af samme dimensioner og type som `X` med et billede svarende til `X` tilføjet en ramme med bredden `width` og farven `colors`.
Formål:	At tilføje en ramme til billedet i `X`.
Filer udleveret:	Ingen.
Begrænsninger:	*Der må kun bruges en løkke (loop), når `X` er et 3D array.*

Algoritme:
Man skal selv designe og implementere denne algoritme.

5. **mirror**-operationen.

Funktions-navn:	`mirror`
Input-parametre:	`X` er et 2D eller 3D array af typen `uint8` med et billede. `w` er bredden af billedet i pixels, dvs. værdien af `dimX(2)`. `lr` er en logisk værdi, som specificerer hvilken vej der skal spejles: — hvis `lr` = `true` skal venstre halvdel spejles til den højre side, — hvis `lr` = `false` skal højre halvdel spejles til den venstre side.
Output-parameter:	`Y` er et array af samme dimensioner og type som `X` med et billede hvor højre/venstre halvdel af `X` er erstattet af spejlbilledet af den anden halvdel.
Formål:	At spejle venstre eller højre side af `X` til den modsatte side.
Filer udleveret:	Ingen.
Begrænsninger:	*Der må ikke bruges løkker (loops) i funktionens krop.*

Algoritme:
Man skal selv designe og implementere denne algoritme. Bemærk at hvis antallet af søjler i billedet er ulige, så skal den midterste søjle ikke ændres.

6. **jail**-operationen.

Funktions-navn:	`jail`
Input-parametre:	Man skal selv specificere de nødvendige parametre.
Output-parameter:	Man skal selv specificere denne parameter.
Formål:	At tilføje lodrette bjælker til `X`.
Filer udleveret:	Ingen.
Begrænsninger:	*Der må kun bruges løkker (loops), når `X` er et 3D array.*

Algoritme:
Resultatet opnås ved hjælp af elementvis multiplikation af hvert af de maksimalt 3 lag af billed-arrayet med et array M af samme højde og bredde. Antag at begge arrays har R rækker og C søjler. Elementerne i M er mellem 0 og 1, og de er givet ved:

$$M_{ij} = \sin(\pi j/p)^2 \ , \quad \text{for} \ \ j = 1, 2, \ldots, C \ \ \text{og} \ \ i = 1, 2, \ldots, R.$$

Bemærk at p er et positivt reelt tal som bestemmer afstanden mellem bjælkerne (jo større p jo længere afstand). Værdien af p specificeres af brugeren.
Tip: Hvis man ønsker at danne et 4x3 array, hvor alle rækkerne indeholder elementerne 7, 2 og 5, kan dette gøres således:

```
ones(4,1) * [7 2 5];
```

7. **superpixel**-operationen.

Funktions-navn:	`superpixel`
Input-parametre:	`X` er et 2D eller 3D array af typen `uint8` med et billede. `dimX` er et 1x3 array indeholdende dimensionerne af `X`. `p` er et heltal der definerer størrelsen af en superpixel (dvs en superpixel består af $p \times p \times$ `dimX(3)` pixels i `X`).
Output-parameter:	Man skal selv specificere denne parameter.
Formål:	At lave en grovere pixeleret version af `X`.
Filer udleveret:	Ingen.
Begrænsninger:	*Brug array-operationer hvor det er muligt.*

Algoritme:
Man skal selv designe og implementere denne algoritme.

Denne funktion beregner en såkaldt "pixelering" af billedet, hvor alle pixels i hvert af de maksimalt 3 lag af en superpixel – hvor hvert lag består af et delbillede af størrelsen $p \times p$ pixels – sættes til middelværdien af pixelværdierne i det samme lag af det tilsvarende delbillede i `X`.

Man skal kun lave pixeleringen i de fuldstændige delbilleder som passer inde i det givne billede. Dvs hvis $p = 3$ og billedet `X` er $10 \times 11 \times$ `dimX(3)`, så er det ok at ignorere den nederste række og de sidste to søjler i `X`.

8. **chess**-operationen.

Funktions-navn:	`chess`
Input-parametre:	`X` er et 2D eller 3D array af typen `uint8` med et billede. `h` og `w` er hhv. højden og bredden af billedet (målt i pixels). `p` er kantlængden af et 2D delbillede (dvs et delbillede består af $p \times p$ pixels).
Output-parameter:	Man skal selv specificere denne parameter.
Formål:	At tilføje et positivt/negativt skakbræt-mønster i billedet `X`.
Filer udleveret:	Ingen.
Begrænsninger:	*Du skal bruge din funktion* `negative`*, men ikke dybden af billedet.*

Algoritme:
Man skal selv designe og implementere denne algoritme.

Denne funktion beregner en variant af pixeleringen af billedet, hvor hver andet delbillede i output-billedet er negativet til det tilsvarende delbillede i input-billedet `X`. De andre delbilleder er uændret. Hvert delbillede er $p \times p$ pixels. Man skal bruge funktionen `negative` til at beregne de negative del-billeder. Også hér kan man ignorere overskydende rækker og søjler i bunden og til højre i billedet.

Specifikation af driver script'et.

Navn på script:	`dims`
Input:	I starten indlæses navnet på en billed-fil i variablen `imagefile`, der således bliver et array of char (string). Herefter indlæses billedet via kald af `imread` ind i variablen `X`, der således bliver et 2D eller 3D array of `uint8`. Resten af indlæsningerne afhænger af de kommandoer, som brugeren anvender.
Output:	Hvis brugeren ønsker det, kan der efter programudførelsen eksistere et figur-vindue med et galleri af de billeder, som er dannet via de otte mulige billed-operationer.
Formål:	At sørge for bruger-interface't.
Filer udleveret:	`dims.m` som indeholder et *ufuldstændigt* script.
Begrænsninger:	Ingen. Dvs. der må gerne laves evt. forbedringer af det ufuldstændige script.

Algoritme:
Den centrale algoritme er skitseret i filen `dims.m`. Men der mangler dele af koden samt nogle kommentarer. Man behøver kun at skrive disse manglende dele, men hvis man ser muligheder for forbedringer, må man gerne ændre i det ufuldstændige script. Vi har forsøgt at tydeligt markere i filen hvor man skal indsætte de manglende dele. Foruden de indledende kommentarer, drejer det sig om at specificere en passende datastruktur for billed-galleriet (to steder i koden) og en betingelses-sætning, der enten udskriver fejlmeddelser, kalder een af de otte funktioner med de nødvendige input-værdier (hvis gyldighed ikke behøver at checkes) eller udfører en af de tre kommandoer `newX`, `showGal` eller `stop`. Effekten af disse tre kommandoer er beskrevet i `dims.m`.

Rapporten

Der skal skrives en *kort rapport* som beskrevet nedenfor. Man opfordres til at skrive en konklusion, hvor man opsummerer hvad man har lært om MATLAB-programmering i projektet.

Rapporten skal indeholde følgende:

- En forside med:
  - navne, studienumre og gruppenummer
  - underskrifter, og
  - databaren hvor man arbejder (hvis man har siddet i en databar).
- En side som viser et billed-galleri i stil med det på side 2, dvs. resultatet af de otte operationer anvendt på hhv. et farvebillede og et gråtone-billede (af eget valg), samt resultatet af kommandoen `newX`, når der indlæses fra en fil eller hentes et billede fra galleriet. For de funktioner der kræver ekstra parametre skal der ovenover billederne stå, hvilke parametre der blev brugt til at lave de viste billeder.
- En beskrivelse af de algoritmer, der er blevet benyttet til at implementere kommandoerne `newX` og `showGal`.
- En listning af det endelige (komplette) script `dims.m` samt de otte MATLAB-funktioner.

Matlab-koden

Du skal følge disse regler i din MATLAB-kode.

- Koden skal virke for alle gråtone- og RGB-farve-billeder, som imread indlæser i arrays af typen `uint8`.
- Variable skal have *selv-forklarende navne*, som gør det nemmere at forstå koden. Hvor det er muligt, skal variabel-navne svare til dem i projektopgaven.
- Jeres kode skal inkludere *kommentarer*, der gør det klart for læseren hvad koden gør. I skal forklare den *generelle ide* bag hvert logisk sammenhængende stykke kode, og kommentere de dele, der trods passende variabel-navne ikke er selv-forklarende ud fra sammenhængen. I skal ikke skrive yderligere kommentarer (f.eks. for hver eneste instruktion), idet en erfaren programmør skal have mulighed for at se sammenhængen i koden ved at springe de for ham/hende unødvendige kommentarer over.
- Der bør anvendes et layout med passende indrykninger af indholdet i betingede sætninger og løkker, og relativt korte linier, der øger læsbarheden af koden.
- Det er **obligatorisk** at skrive kommentarer i starten af hver funktion (såkaldt "help text") som beskriver formålet med funktionen (hvad den 'gør') i den første linie, input- og output-parametrene samt giver et eksempel på hvordan funktionen kaldes.

  Eksempler på "help text" er givet i `grayscale.m`. Man kan også finde inspiration i MATLAB's funktioner: i kommando-vinduet kan man skrive `help` efterfulgt af et funktionsnavn (fx `help sin` eller `help grayscale`); hvad man så ser, er "help-teksten" for funktionen ned til den første linie, der ikke starter med '%'-tegnet.

  Som I ved er "help text" også beskrevet i lærebogens afsnit 10.3.5.
- Man kan konvertere `uint8`-værdier til tilsvarende `double`-værdier via funktionen `double`, ligesom man kan afrunde `double`-værdier til nærmeste `uint8`-værdi via funktionen `uint8`. Man bør dog *kun* konvertere disse værdier *hvis det er nødvendigt*!