Rörelsedetektion i bilder

Signaler och system

ht02

Uppsala universitet

Dahlin Gustav	781227-0176
Persson Andreas	800501-6616
Severinson Joakim	770712-4611
Johansson Pär	810429-9071

Sammanfattning:

Projektet handlar om att detektera förflyttningar i bilder. Att detektera en förflyttning av ett objekt mellan två bilder, representera förflyttningen på ett överskådligt och begripligt sätt, visar sig bestå av en hel del svåra problem. Problem såsom stora skillnader i ljusintensitet orsakade av skuggor och reflektioner, samt små men betydande avvikelser mellan bilderna orsakat av förskjutning av kameran vid bildtagning måste hanteras.

Korskorrelationen beräknas efter eventuell filtrering för att detektera rörelser och riktning. För att presentera resultatet på ett överskådligt och begripligt sätt tas en riktningsvektor fram för att visa i vilken riktning objektet har rört sig. Det som inte lyckas lösas på ett tillfredsställande sätt är problem med förskjutning i bilder.

Innehållsförteckning

Inledning och problembeskrivning

Ett av dom stora forskningsområdena nu är datoriserad bildanalys, där man vill datorisera förståelse och utbringandet av data ur bilder. Man vill komma åt detaljer och konturer i bilder som inte är tydliga nog för oss att uppfatta, eller analysera utan hjälp, man vill även koppla funktionalitet och automatik till bilder. Datoriserad bildanalys har växt under kort tid vilket till stor del beror på att det krävs väldigt mycket datorkapacitet för att jobba med bilder. Den möjligheten har inte funnits i samma utsträckning tidigare.

Projektet bygger på att detektera rörelser i bilder innehållande liknande objekt. Tanken är att detta kan användas för att t.ex. automatisera rörelsedetektion i kameror som sedan kan mäta hastigheten hos objektet. Eftersom kameror bygger på att ta bilder med korta intervaller borde bildbehandling kunna detektera skillnader och således även rörelser mellan två eller fler bilder.
För att detektera ett objekts rörelse mellan två tagna bilder måste projektet klara av ett antal kriterier:

- Arbeta med både bilder i färg och gråskala.
- Hantera olika bakgrundsintensiteter
- Säkerställa huruvida ett objekt i en bild är densamma i den andra bilden
- Rörelse i tre dimensioner.
- Rotation av ett objekt.
- Särskilja rörelser av flera objekt i samma bild.
- Hantera ljusfenomen som kan uppstå i bilder, t.ex. skuggor, reflektioner etc.
- Mäta rörelsens sträcka.

Några av dessa kriterier ryms inte inom detta projekt, några beroende på att de inte går att lösa med den kunskap som kursen förmedlar, andra av att de inte ryms inom projektets omfång.
De kriterier projektet hanterar är

- Säkerställa huruvida ett objekt i en bild är detsamma i en andra bild
- Hantera ljusfenomen som kan uppstå i bilder, t.ex. skuggor, reflektioner etc.
- Rörelse i två dimensioner.[1]
- Endast ett objekt per bild som förflyttar sig är möjligt för att detektering ska kunna ske.

Huvudproblemet med rörelsedetektionen är att samtidigt som avvikelser mellan bilder måste hittas skall det även detekteras att dessa avvikelser är samma objekt i båda bilderna.
Detta bildar kärnan i projektet.

[1] Rörelse I Z-led, dvs. i bildens djup, ger problemet en helt ny karaktär.

Problemlösning och metod

Bilder som signaler

Digitala bilder byggs upp av pixlar i x och y-led där varje pixel innehåller information om intensitet. För en bild i gråskala behövs bara ett lager av pixlar som beskriver t.ex. nivån av svart i varje pixel.
Bilder i färg är en blandning av de tre komponenterna röd, grön och blå (RGB). Alltså måste varje pixel i en färgbild innehålla intensitetsinformation för samtliga RGB-komponenter.
För att arbeta med bilder som signaler måste dessa hanteras som en signal som sträcker ut sig i två dimensioner, med andra ord har bilder spektrum i både x och y-led, med amplituden representerad i en tredje dimension.
För bilder i gråskala kan således signalrepresentationen vara en tvådimensionell matris med samma storlek som antalet pixlar i x och y-led. För färgbilder måste varje RGB-index delas upp, och resulterar i en tredimensionell matris.

Korskorrelation

Det finns ett flertal metoder för att bestämma skillnader mellan bilder, varav de flesta endast analyserar avvikelsen. En metod för att bestämma skillnader, framför allt likheter mellan bilder, är korskorrelationen. Korskorrelationen definieras som en faltning:

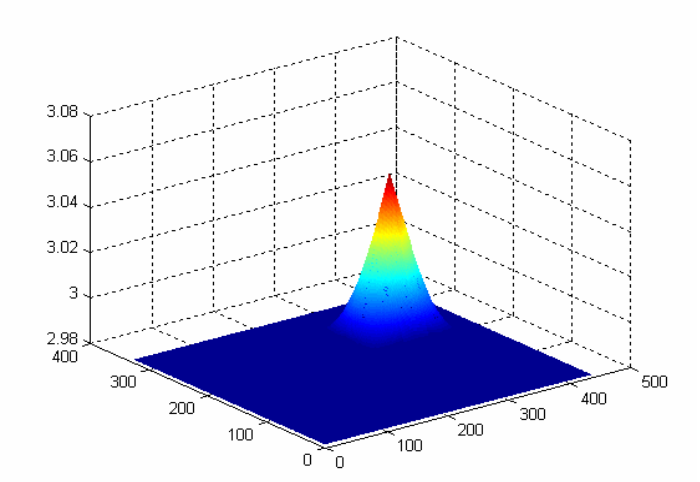
$r_{xy}(t) = \int_{-\infty}^{\infty} x(t+t)y(t)dt$, där x och y är signaler.

Korskorrelationen för bilder blir, trots skillnaden i dimensioner, definitionsmässigt inte annorlunda. Den definieras fortfarande som en faltning mellan två signaler, vilka representeras av två matriser.
Korskorrelationen används på så sätt att den ena bilden får agera matchande filter medan den andra ses som en ren insignal som ska passera det matchande filtret. Faltningen definieras då som:

$y(t) = \int_{-\infty}^{\infty} h(t-t)x(t)dt$,

där h är den förskjutna bilden som fungerar som matchande filter[2] och x är den andra bilden tolkad som insignal.

Utsignalen, *y(t)*, är ett mått på likheten mellan filtret h och insignalen x, d.v.s. likheten mellan bilderna baserat på den bild som agerar filter. Som utsignal fås en matris med samma dimension som *x(t)*. En graf över korskorrelationen mellan två bilder ses i figuren till höger



[2] Bilden h måste spegelvändas I x,y-led för att åstadkomma förskjutningen.

Detektion och beräkning av förflyttning

Resultatet av korskorrelationsbräkningen används för att bestämma två egenskaper, huruvida det finns två objekt som är samma i bilderna, om så är fallet så kan förflyttningsvektorn beräknas.
I figuren i sektionen ovan ses att ett tydligt maximum kan utläsas ur en korskorrelationsberäkning mellan två (för figuren: ideala) bilder. Det räcker dock inte att hitta maximum för korrelation, det måste även verifieras att korrelationsnivån är hög nog för att betraktas som en förflyttning. En enkel verifiering är att bestämma ett tröskelvärde för förhållandet mellan max och medelnivå hos korrelationen. Är förhållandet högre än tröskelvärdet anses det ha skett en förflyttning av ett och samma objekt mellan bilderna.
Korskorrelationen säger normalt något om var i insignalen likheten finns. Detta gäller dock bara för matchande filter som har ett impulssvar som endast motsvarar den sökta signalen, det matchade filtret representeras nu av en hel bild där endast en mindre del är den signal vi matchar mot.
Resultatet av detta blir att korskorrelationen i detta fall inte kan ses som en positionsangivelse för var i insignalen det liknande objektet befinner sig. Det som dock går att finna, om korrelationen är bra nog, är den absoluta förflyttningen i pixlar som objektet gjort mellan bilderna.
För att kunna ta fram den absoluta förflyttningen måste en extra parameter vara känd, nämligen i vilken riktning objektet har förflyttat sig i x och y-led, riktningsvektorn. Riktningsvektorn tas fram genom att räkna fram differensbilden mellan bilderna och sedan ta fram vektorn genom subtraktion av minimumpositionen och maximumpositionen i differensbilden. Det kan även behövas viss manipulation av differensbilden, beroende på brusskillnader, för att få fram bra max. och min. positioner.
För vissa differensbilder blir resultatet väldigt brusigt, vilket leder till stora fel i riktningsvektorn. Detta löses genom att beräkna medelvärdet av differensbilden och undertrycka komponenter som är högre och lägre än någon lämplig konstant multiplicerat med medelvärdet.

Bildanalysproblem.

Ofta finns det problem som måste tas hand om innan själva rörelsedetekteringen kan ta vid. Vid optimala bilder blir oftast inte rörelsedetektionen något problem, men om t.ex. något eller flera av nedanstående effekter finns i en eller båda bilderna så uppstår problem med detektionen.

- Förskjutning av ena bilden – Kameran är inte fixerad
  - Ett mycket svårt problem för korskorrelationen att hantera eftersom nästan allt i insignalen kommer få en hög korrelationsgrad i förhållande till det matchade filtret, eftersom bilden uppfattas som en ren skiftning av det matchade filtret. Detta får som konsekvens att det inte går att detektera ett objekt som rör sig mellan två sådana bilder med korskorrelationen. Det enda som skulle gå är att filtrera insignalsbilden så att förskjutningen försvinner.
- Reflektioner och temporära ljusskillnader - Uppkommer vid användning av blixt eller annan varierande ljuskälla när bilderna tas.
  - Det enda som kommer att skilja bilderna undantaget objektet som flyttar sig är ljusintensiteten. Korskorrelationen kommer att uppfatta ljusintensiva areor i de olika bilderna som samma objekt, även om så inte är fallet.
  - Detta tas bäst bort genom olika varianter av intensitetsfiltrering.
  - Filtrering som används här är stark band eller högpassfiltrering för att endast få kvar konturer som är intressanta.

Metoder för lösning.

För att lösa uträkningen av korskorrelationen av två bilder skulle det teoretiskt gå att beräkna detta som en vanligt faltning. Praktiskt är detta dock inte möjligt då det skulle leda till beräkningar som tar lång tid.
Korskorrelationen kan fouriertransformeras så att faltningen hanteras som en multiplikation i frekvensdomän. För att sedan få fram den faltningen inverstransformeras produkten.
Då insignalerna är 2-dimensionella matriser används FFT i två dimensioner för att på snabbast sätt klara av faltningen.
Formel för korskorrelationen implementerad med FFT:

$$r_{hx} = h(t-t) * x(t)$$
$$R_{hx} = FFT(h(t-t))FFT(x(t))$$
$$r_{hx} = FFT^{-1}(R_{hx})$$

Där $h(t-t)$ är den ena bilden flippad som ett matchat filter och *x(t)* den andra bilden som en insignal.

Att tänka på när det gäller bilder är att bilder representeras som matriser så det måste vara elementmultiplikationer istället för matrismultiplikationer.
Korskorrelationen i sig är svår att analysera eftersom den innehåller komplexa värden som genereras från transformeringen. I vissa avseenden är det bättre att titta på absolutbeloppet av korskorrelationen i vissa avseenden.

Resultat

Genom att prova olika typer av bilder ges att korskorrelationen ger bra värden för bilder som endast innehåller ett, lätt urskiljbart, objekt som rör på sig.
Finns det fler förflyttningar i bilden kommer metoden med korskorrelation leda till att antingen flera objekt hittas, vilket leder till fel rörelsedetektering, eller så uppfattas hela insignalsbilden som korrelerad med det matchade filtret. Detta ger undertryckning av en eventuell riktig korrelering. Nedan följer tre typer av bilder där korskorrelationen ger ett önskat resultat. För mer komplicerade bilder, t.ex. förskjutna bilder kan metoden med korskorrelation och enkla filter inte användas för att få ett önskat resultat.

Idealt konstruerade bilder.

Bild1 används som det matchade filtret.
En idealbild, RGB-intensiteterna representeras lika mycket i bakgrunden, objektets intensitet är skarp gentemot bakgrunden

Bild2 används som insignalen *x(t)*.
Detta är också en idealbild, objektet är precis det samma som i Bild1 fast förflyttat

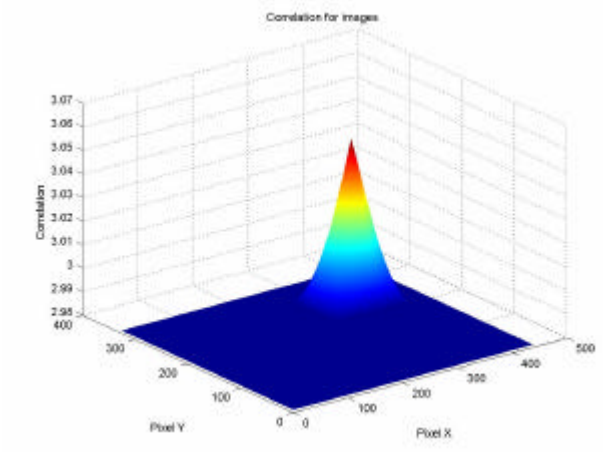
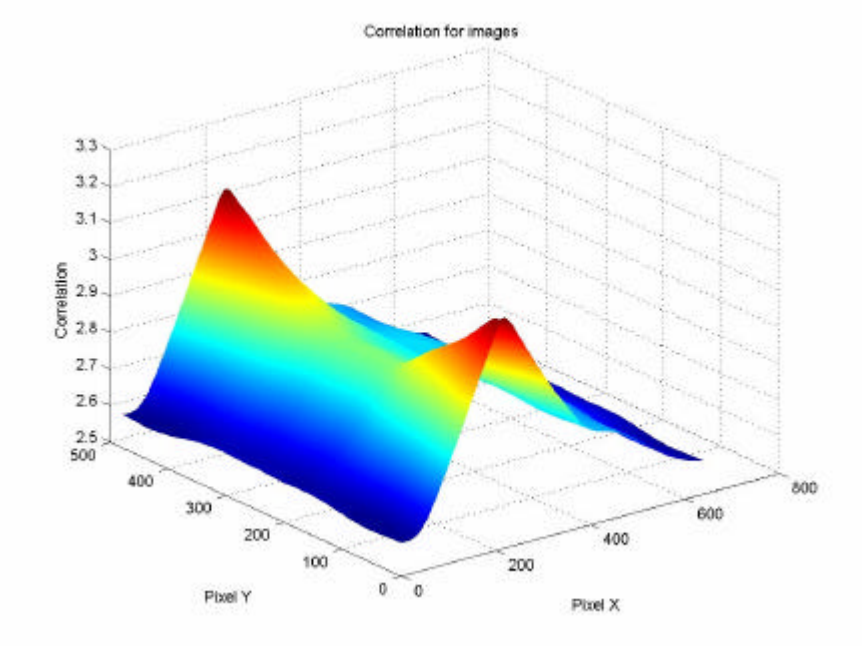


Bild1 och bild 2 som inparametrar ger en ideal korrelation, som ses ovan, att beräkna förflyttningen ur.

Bilden till höger representerar förflyttningen av objektet. Förflyttningen representeras på flera sätt, färger anger om det är en startposition eller ny position, förflyttningen i pixlar ges av axlarna x och y och vi representerar i vilken riktning objektet har förflyttat sig.

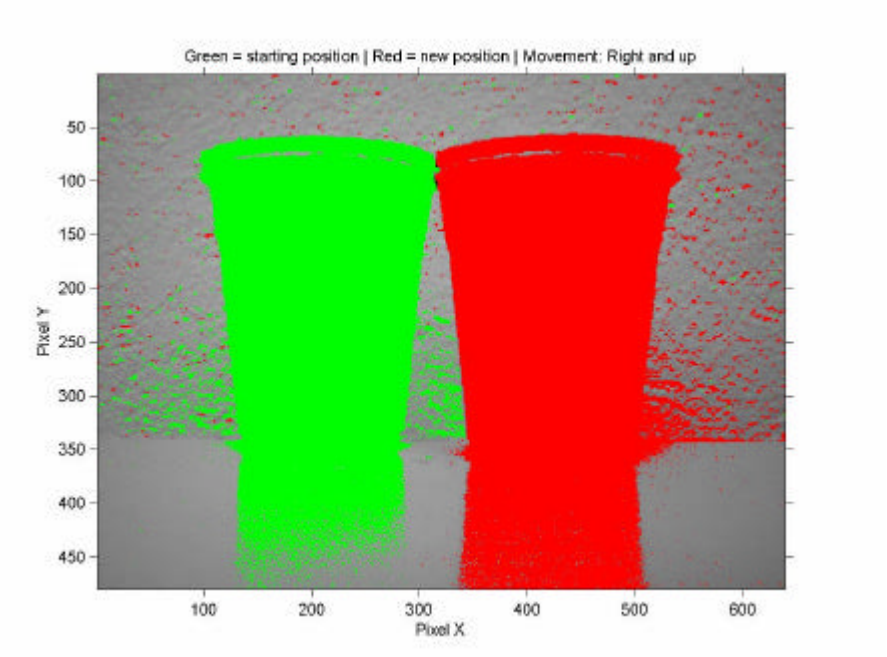
Bra fotograferade bilder

Bilderna till vänster innehåller en kaffemugg fotograferad på väldigt nära avstånd. Den har förflyttat sig åt höger, dock innehåller de knappt har några andra skillnader såsom ljusskillnader etc.



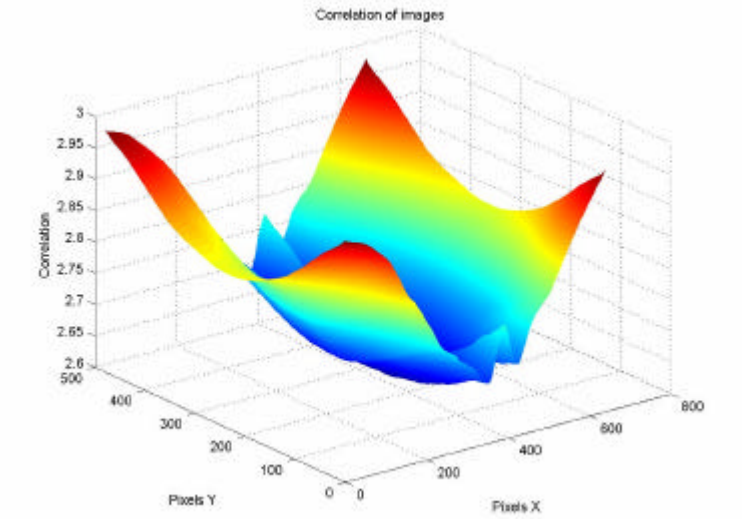
Korrelationen är tydlig och lätt att utläsa ur grafen.

I bilden till höger syns förflyttningen som är tydlig.

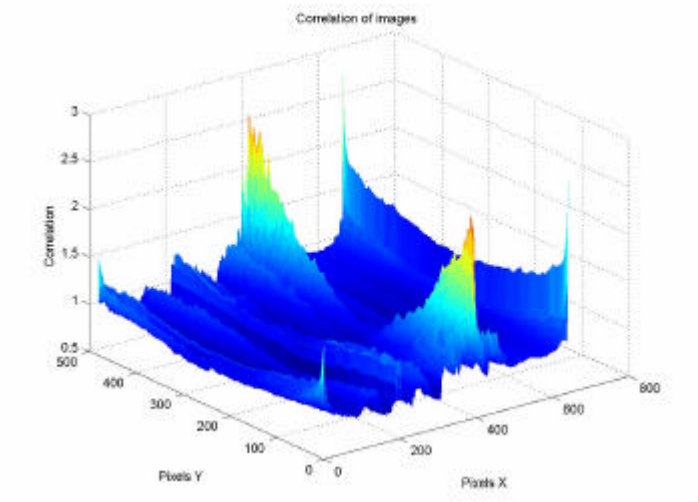


Bilder med ljus och temporära ljusskillnader


I bilderna ovan har burken flyttat sig en liten bit till höger, men vad som kanske inte uppfattas vid första blicken är att bilderna skiljer sig ganska markant vad det gäller ljusintensitet på olika ställen. Detta ses genom att korrelationen nedan får sina högsta värden ute i hörnen, d.v.s. den högsta korrelationen ges för objekt som flyttat sig endast ett apr pixlar. Så är inte fallet och man kan se den riktiga korrelationen i mitten på grafen.



För att finna den riktiga korrelationen filtreras bilderna med ett kraftigt bandpassfilter och där följande korrelation blir resultatet:



Vi har förstärkt fram den korrelation vi vill ha och kan finna rörelsen mellan bilderna.

Slutsats och diskussion

Av resultatet ses att korskorrelation är en bra metod att använda för att detektera rörelser i bilder. Korskorrelationen funkar väldigt bra så länge det bara är ett objekt i bilderna som rör på sig, har vi flera objekt, oavsett typ, kommer korskorrelationen att registrera dessa. Registrerar korskorrelationen flera objekt gäller det att kunna applicera ett filter för dessa rörelser som kan vara allt ifrån riktiga rörelser till ljusintensiteter.
Dock räcker det inte bara med korskorrelation för att klara av att hantera alla typer av bilder som kan förekomma. För att klara av t.ex. flera rörliga objekt samt andra förändringar i bilder såsom t.ex. ljusskillnader och rörelse i z-led krävs andra metoder i kombination med korskorrelationen.
För att använda korskorrelation i många områden räcker det med filtrering som tar bort eller undertrycker oönskade störningar till önskad grad.

Utvidgning av projektet skulle kunna göras genom att kunna plocka ut ett objekt via t.ex. differensbilden och sedan göra korskorrelation med detta objekt som matchande filter istället för en hel bild. Några andra förbättringar är att låta göra bättre bildanalys för att automatiskt kunna applicera ett bra filter innan korskorrelation görs.

Appendix

---

main.m

```
% The main function that should be executed.

% Clear windows and things...
clc;
clear;
close all;

% Read images into matrixes
Xs = imread('bilder\sprite1.jpg');
MF = imread('bilder\sprite2.jpg');

% Filter images...
[fXs,fMF]= ourFilter(Xs, MF);       % If we want to filter images.
%fXs = Xs; fMF = MF;                % If we DONT want to filter images.

% Change to double from im2
Xs = im2double(fXs);
MF = im2double(fMF);

threshCCorr = 0.99;        % The threshold for the cross correlation in percent.
threshMove = 0.15;         % The threshold for the matching of direction.

diffIm = diffImages(MF, Xs);    % The differens b/w images

% Get the direction and the targetpoints
move = movement(Xs, MF, diffIm, threshCCorr, threshMove);

lower = diffIm < -0.2;      %  ones for values less than -0.2
upper = diffIm > 0.2;       %  ones for values greater than 0.2

% Add the colours (red, greeen) to the image
gray = 0;
rgb = 0;
for i=1:3
   gray = gray + ((Xs(:,:,i).* not(lower)) .* not(upper))/3;
end
rgb = cat(3, (gray + double(upper)), (gray + double(lower)), gray);

figure, imshow(rgb), title(['Green = starting position', ' | ', 'Red = new position', ' | Movement: ',
movementToString(move(1))]);
axis on;
```

maxCCorr.m

```
% Cross correlation function
function result = maxCCorr(th, picXs, picMF)

cCorr = 0;

%Loop for all 3 RGB-layers
for i=1:3
   %normalize matrices
   Xs = picXs(:,:,i)./norm(picXs(:,:,i));
   MF = picMF(:,:,i)./norm(picMF(:,:,i));

   %Flip the MF-matrix
   MF = flipud(MF);
   MF = fliplr(MF);

   x = fft2(MF);
   y = fft2(Xs);
   cCorr = cCorr + x.*y;
end

% Inverse Transformation of the correlation matrices
cCorr = ifft2(cCorr);
cCorr = abs(cCorr);

% Print out the correlation graph
figure, surf(cCorr), shading flat

% Mean of the correlation & the maximum correlation
maxInfoCCorr = maximum(cCorr);
meanLevel = mean(mean(cCorr));

% Calculate level of correlation related to the mean value of correlation
cCorrLevel = meanLevel/maxInfoCCorr(1);

%If correlation is large enough, return correlation matrix, otherwise an
%error matrix
if(cCorrLevel < th)
   result = maxInfoCCorr;
else
   result = [0, -1, -1];
end
```

movement.m

```
% Returns the movement of an object with return values as following:
% direction (1-4), % x-start-position, % y-start-position, %
% x-end-position, % y-end-position

function result = movement(Xs, MF, diffIm, threshCCorr, threshMove)

% Get the values greater than 0 and less then zero
a = diffIm > 0;
b = diffIm < 0;

% Pick out the values from the differens matrix
positive = diffIm .* a;
negative = diffIm .* b;

% Get the min/max position.
minPos = maximum(chop(negative));
maxPos = maximum(chop(positive));

% Compute Cross Correlation.
maxCCorrPos = maxCCorr(threshCCorr, Xs, MF);

% Check if value if valid. (0,-1,-1) means to small threshold for
% correlation.
if maxCCorrPos ~= (0 -1 -1)
   % Get the size of the images.
   imageSize = size(Xs);

   % Get all possible directions.
   vDirection = directions(maxPos, minPos, imageSize);

   % Match the correlation with the possible directions.
   d = direction(vDirection, maxCCorrPos, imageSize.*threshMove);

   result = [d(3), minPos(2), minPos(3), maxPos(2), maxPos(3)];
else
   result = [-1, minPos(2), minPos(3), maxPos(2), maxPos(3)];
end
```

directions.m

```
% Returns the possible movements:
%Index      Movement
% 1.        Right and down.
% 2.        Right and up.
% 3.        Left and down.
% 4.        Left and up.
function result = directions(maxpos, minpos, imageSize)

% The differens in x and y
dx = abs(minpos(2) - maxpos(2));
dy = abs(minpos(3) - maxpos(3));

% Construct the possible movements matrix
result = [dx                dy,
        dx                (imageSize(1) - dy),
        (imageSize(2) - dx)   dy,
        (imageSize(2) - dx)   (imageSize(1) - dy)];
```

---

direction.m

```
% Returns the distances (in x and y) and the direction of the movement (1-4).
function result = direction(vDirection, maxCCorr, thresh)

dir = -1;

% Find out if one of the possible movements matches with the correlation.
for i = 1:4,
   if ((abs(vDirection(i, 1) - maxCCorr(2)) < thresh(1)) && (abs(vDirection(i, 2) - maxCCorr(3)) <
thresh(2)))
      dir = i;
      break;
   end
end

% If we found a direction return it, otherwise return 0
if(dir > 0)
   result = [vDirection(dir, 1), vDirection(dir, 2), dir];
else
   result =[0, 0, 0];
end
```

chop.m

```
% Chop down a matrix by a given value
function result = chop(matrix)

temp = abs(matrix);         % Absolute value of each cell in matrix
s = size(temp);             % Size of the matrix
m = 3 * mean(mean(temp));   % Get the mean value * magic value

for y = 1:s(1)
   for x = 1:s(2)
      if (temp(y,x) > m)
         temp(y,x) = m;
      end
   end
end

% figure, surf(temp), shading flat;
result = temp;
```

diffImages.m

```
% Computes the differens between two RGB images.
function result = diffImages(im1, im2)

d = 0;

% Make a sum of the difference for all of the indexes (RGB).
for i = 1:3
   d = d + (im1(:,:,i) - im2(:,:,i));
end
% Returning value
result = d;
```

maximum.m

```
% Returns the maximum value in the given matrix and its position.
function B = maximum(A)
temp = [A(1,1),1,1];
sizeA = size(A);
% Go through the whole matrix...
for y = 1:sizeA(1)
   for x=1:sizeA(2)
      if (A(y,x) > temp(1))
         temp(1) = A(y,x);
         temp(2) = x;
         temp(3) = y;
      end
   end
end

% Returning values
B = temp;
```

minimum.m

```
% Returns the minimum value in the matrix and ints position.
function B = minimum(A)

temp = [A(1,1),1,1];
sizeA = size(A);

% Go through the whole matrix...
for y = 1:sizeA(1)
   for x=1:sizeA(2)
      if (A(y,x) < temp(1))
         temp(1) = A(y,x);
         temp(2) = x;
         temp(3) = y;
      end
   end
end

B = temp;
```

---

movementToString.m

```
% Converts the movement index to a string.
function result = movementToString(index)

switch(index)
   case 1
      s = 'Right and down';
   case 2
      s = 'Right and up';
   case 3
      s = 'Left and down';
   case 4
      s = 'Left and up';
   otherwise
      s = 'No movement found';
end

result = s;
```

ourFilter.m

```
% Filters the two matrixes with a Butterworth bandpass filter
% Returns the two filterd matrixes
function [result1,result2] = ourFilter(Xs, MF)

% size of matrixes
s = size(Xs);

% Butterworthfilter
h = butter(1, [0.30 0.40]);

% Applying the filter to the matrixes
tempXs = imfilter(Xs, h);
tempMF = imfilter(MF, h);

% Returning the matrixes
result1=tempXs;
result2=tempMF;
```