

**2020**

**Δ.Π.Μ.Σ. ”ΣΥΣΤΗΜΑΤΑ ΕΠΕΞΕΡΓΑΣΙΑΣ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΜΗΑΧΑΝΙΚΗ ΝΟΥΜΟΣΥΝΗ”**

Ψηφιακή Επεξεργασία Βίντεο

1η Εργαστηριακη ασκηση:

Οπτική Ροή για Σταθεροποίηση Βίντεο

χατζηκαλυμνιοσ ευαγγελοσ ΑΜ: (1003867)

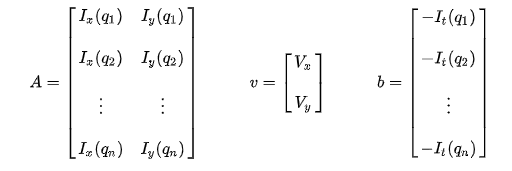
# Οπτική Ροή Lucas-Kanade

Υλοποιώντας τον αλγόριθμο οπτικής ροής χρησιμοποιώντας τη μέθοδο Lucas-Kanade υπολογίζουμε κίνησης u(Vχ,Vy) για κάθε pixel. Συγκεκριμένα, τα δια-νύσματα πρέπει να ικανοποιούν τις παρακάτω σχέσεις:

Εικόνα που περιέχει μαχαίρι, πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα (1)

Όπου Ix , Iy και It οι μερικές παράγωγοι ως προς τις γραμμές, τις στήλες και τον χρόνο αντίστοιχα. Το w ορίζει το μέγεθος του παραθύρου στο οποίο γίνονται οι παραπάνω υπολογισμοί, θεωρώντας ότι μέσα σε αυτό η ταχύτητα είναι σταθερή. Οι παραπάνω σχέσεις γράφονται σε μορφή πινάκων ως , όπου:

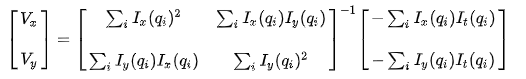


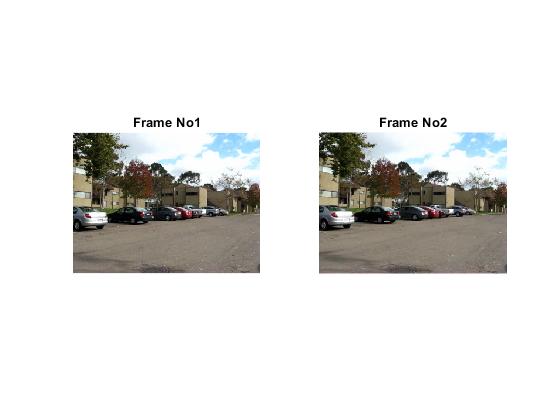
Και τελικά το σύστημα γίνεται:

Εικόνα που περιέχει αντικείμενο

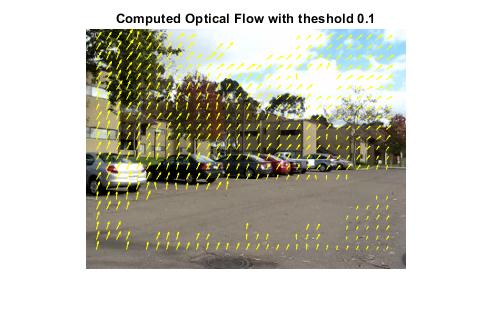
Περιγραφή που δημιουργήθηκε αυτόματα (2)

Όπου ο ανάστροφος πίνακας και o second moment matrix. Αναλύοντας από τη σχέση (2) προκύπτει ότι:

 (3)

  *Εικόνα που περιέχει άνδρας, γάτα, λευκό

Περιγραφή που δημιουργήθηκε αυτόματα*



Εικόνα που περιέχει φωτογραφία, δέντρο, άνδρας, πίνακας

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει φωτογραφία, δέντρο, άνδρας, πίνακας

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει φωτογραφία, άνδρας, πίνακας, νεαρός

Περιγραφή που δημιουργήθηκε αυτόματα

2) Ανίχνευση Γωνιών

Για την ανίχνευση ακμών γίνεται έλεγχος στις ιδιοτιμές του μητρώου . Συγκεκριμένα ορίζεται ένα κατώφλι με το οποίο συγκρίνονται οι ιδιοτιμές. Αν και οι δυο έχουν μεγάλη τιμή, μεγαλύτερη από το κατώφλι που έχουμε ορίσει θεωρούμε ότι το συγκεκριμένο pixel αντιστοιχεί σε γωνία. Αντίστοιχα μπορούμε να υπολογίσουμε την ποσότητα

(4)

όπου k μία εμπειρική σταθερά μεταξύ του 0.04 και 0.06. Η παραπάνω ποσότητα R συγκρίνεται με ένα κατώφλι και οι τιμές που το υπερβαίνουν είναι οι πιθανές γωνίες της εικόνας. Επειδή ανάλογα με το παράθυρο w που επιλέγουμε, η ίδια γωνία μπορεί να εμφανίζεται σε πολλά σημεία του παραθύρου, επιλέγουμε μόνο τις μεγαλύτερες τιμές του R εντός του παραθύρου.

Εικόνα που περιέχει οδός, φωτογραφία, σταθμευμένος, στάθμευση

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει φωτογραφία, οδός, δέντρο, άνδρας

Περιγραφή που δημιουργήθηκε αυτόματα

## 3) Αραιή οπτική ροή

Με βάση τα παραπάνω βήματα μπορούμε να υπολογίσουμε την οπτική ροή μόνο στις γωνίες της εικόνας και όχι σε όλη την εικόνα. Υπολογίζοντας λοιπόν τις γωνίες βρίσκουμε την οπτική ροή στα παράθυρα με κέντρο τις γωνίες.

## Εικόνα που περιέχει κίτρινο, οδός, ιππασία, άνδρας Περιγραφή που δημιουργήθηκε αυτόματα

## 4) Σταθεροποίηση βίντεο

## Α. λαμβάνοντας υπόψη την οπτική ροή από frame σε frame

Με βάση τις ταχύτητες που έχουνε υπολογιστεί μπορούμε να κάνουμε μία αντιστοίχηση των δύο διαδοχικών εικόνων. Συγκεκριμένα ένα σημείο που βρισκότανε στην θέση [x, y] με βάση την ταχύτητα που υπολογίστηκε στο επόμενο frame θα βρίσκεται στην θέση [x + u, y + v]. Ο μετασχηματισμός γίνεται ανά pixel λαμβάνοντας υπόψη την οπτική ροή από frame σε frame χρησιμοποιώντας τη συνάρτηση interp2 της MatLab. Συνάρτηση αυτή υπολογίζει την παρεμβολή αναμεσά στις θέσεις [x, y] και [x + u, y + v]. Το σταθεροποιημένο βίντεο που προκύπτει αποθηκεύεται με όνομα stabilized\_optical.avi

## Β. Με βάση τις γωνίες και με χρήση μοντέλου affine

Σε αυτή τη περίπτωση για τη σταθεροποίηση του βίντεο εξάγουμε τις γωνίες σε κάθε frame. Συγκεκριμένα Μια γωνία που βρισκόταν στην θέση [x, y] με βάση την ταχύτητα που υπολογίστηκε στο επόμενο frame θα βρίσκεται στην θέση [x+u, y+v]. Άρα έχοντας αρκετά σημεία-γωνίες μπορούμε να υπολογίσουμε τον μετασχηματισμό affine ώστε:

Εικόνα που περιέχει κείμενο, ρολόι

Περιγραφή που δημιουργήθηκε αυτόματα

Κάνοντας αντιμετάθεση στις πράξεις προκύπτει:

Εικόνα που περιέχει ρολόι

Περιγραφή που δημιουργήθηκε αυτόματα

Η λύση του συστήματος με την έννοια των ελαχίστων τετραγώνων προκύπτει από τον ψευδοαντίστροφο του πίνακα A και είναι η εξής



Αφού υπολογίσω το μητρώο μετασχηματισμού κανω χρήση των συναρτήσεων affine2d και imwarp της MatLab για να εξάγω το σταθεροποιημένο frame. Το σταθεροποιημένο βίντεο που προκύπτει αποθηκεύεται με όνομα stabilized\_affine.avi

## **ΠΑΡΑΡΤΗΜΑ:** ΚΩΔΙΚΑΣ ΣΤΗΝ MATLAB

Υλοποιήθηκαν οι 3 βασικές συναρτήσεις και πέντε scripts στην Matlab:

**Functions:**

* function [u,v] = opticalFlow(im1,im2,windowSize)

Υπολογισμός διανυσμάτων οπτικής ροής.

* function [corner\_i, corner\_j] = cornerDetection(im,windowSize, thres)

Υπολογισμός των Pixel που αντιστοιχούν σε γωνίες.

* function I\_warp = WarpImage(I,u,v)

Wrapping ενός frame βάσει της παρεμβολής αναμεσά στις θέσεις [x, y] και [x + u, y + v].

**Scripts:**

* OpticalFlow.m : Σχεδίαση Dense Optical Flow του βίντεο.
* CornerDetection.m : Σχεδίαση γωνιών του βίντεο.
* CornerOpticalFlow.m : Σχεδίαση Optical Flow μόνο για τις γωνίες.
* VideoStabilization.m :Σταθεροποίηση βίντεο με βάση τα διανύσματα [u,v].
* AffineStabilization.m : Σταθεροποίηση βίντεο βάσει μοντέλου affine.

**Functions:**

**1)**

function I\_warp = WarpImage(I,u,v)

% I - image to warp

% u,v - the optical flow parameters gotten from the LK algorithm

[x, y] = meshgrid(1:size(I,2),1:size(I,1));

% interpolate

I\_warp = interp2(I, x+u, y+v, 'cubic');

% in case of NaN values - put the original values of I instead

I\_warp(isnan(I\_warp)) = I(isnan(I\_warp));

end

**2)**

function [u,v] = opticalFlow(im1,im2,windowSize)

w = windowSize;

%offset

off = floor(w/2)+1;

% The Mask

dx = [-1 0 1; -1 0 1; -1 0 1];

dy = dx';

% Lucas Kanade

% for each point, calculate I\_x, I\_y, I\_t

Ix\_m = conv2(im1, dx, 'valid'); % partial on x

Iy\_m = conv2(im1, dy, 'valid'); % partial on y

It\_m = conv2(im1, ones(2), 'valid') + conv2(im2, -ones(2), 'valid'); % partial on t

u = zeros(size(im1));

v = zeros(size(im2));

% within window w \* w

for i = off+1:size(Ix\_m,1)-off

for j = off+1:size(Ix\_m,2)-off

Ix = Ix\_m(i-off:i+off, j-off:j+off);

Iy = Iy\_m(i-off:i+off, j-off:j+off);

It = It\_m(i-off:i+off, j-off:j+off);

Ix = Ix(:);

Iy = Iy(:);

b = -It(:); % get b here

A = [Ix Iy]; % get A here

%second moment matrix

M = A.'\*A;

%eigenvalues of matrix M

e = eig(M);

%eigenvalues threshold

thresh = 0.3;

if (e(1) < thresh) || (e(2)<thresh)

u(i,j)= 0;

v(i,j)= 0;

else

nu = pinv(M)\*A.'\*b; % get velocity here

u(i,j)=nu(1);

v(i,j)=nu(2);

end;

end;

end;

end

**3)**

function [corner\_i, corner\_j] = cornerDetection(im,windowSize, thres)

w = windowSize;

%offset

off = floor(w/2)+1;

dx = [-1 0 1; -1 0 1; -1 0 1]; % The Mask

dy = dx';

% Lucas Kanade

% for each point, calculate I\_x, I\_y, I\_t

Ix\_m = conv2(im, dx, 'valid'); % partial on x

Iy\_m = conv2(im, dy, 'valid'); % partial on y

corner\_i = [];

corner\_j = [];

% within window w \* w

for i = off+1:size(Ix\_m,1)-off

for j = off+1:size(Ix\_m,2)-off

Ix = Ix\_m(i-off:i+off, j-off:j+off);

Iy = Iy\_m(i-off:i+off, j-off:j+off);

Ix = Ix(:);

Iy = Iy(:);

A = [Ix Iy]; % get A here

%second moment matrix

M = A.'\*A;

%eigenvalues of matrix M

e = eig(M);

%R(i,j) = det(M)-.05\*trace(M)

if (e(1)>thres) && (e(2)>thres)

corner\_i = [corner\_i i];

corner\_j = [corner\_j j];

end

end;

end;

end

**scripts**

1. **OpticalFlow.m**

close all;

clear all;

%Read a video

v = VideoReader('video1.avi');

%Extract Frames from video

video = read(v);

%Frame No1

fr1 = video(:,:,:,1);

subplot 121

imshow(fr1);

title('Frame No1', 'FontSize', 10);

%Frame No2

fr2 = video(:,:,:,2);

subplot 122

imshow(fr2);

title('Frame No2', 'FontSize', 10);

% downsize to half

im1t = im2double(rgb2gray(fr1));

im1 = imresize(im1t, 0.5);

im2t = im2double(rgb2gray(fr2));

im2 = imresize(im2t, 0.5);

windowSize = 9;

[u,v] = opticalFlow(im1,im2,windowSize);

% downsize u and v

u\_deci = u(1:5:end, 1:5:end);

v\_deci = v(1:5:end, 1:5:end);

% get coordinate for u and v in the original frame

[m, n] = size(im1t);

[X,Y] = meshgrid(1:n, 1:m);

X\_deci = X(1:10:end, 1:10:end);

Y\_deci = Y(1:10:end, 1:10:end);

figure();

imshow(fr2);

hold on;

% draw the velocity vectors

quiver(X\_deci, Y\_deci, u\_deci,v\_deci, 'y')

title('Computed Optical Flow with theshold 0.3', 'FontSize', 10)

1. **CornerDetection.m**

close all;

clear all;

%Read a video

v = VideoReader('video1.avi');

%Extract Frames from video

video = read(v);

%Frame No1

fr1 = video(:,:,:,1);

im = im2double(rgb2gray(fr1));

%im1 = imresize(im1t, 0.5); % downsize to half

%eigenvalues threshold

thresh = 20;

windowSize = 8;

[corner\_i,corner\_j] = cornerDetection(im,windowSize, thresh);

Image\_corners = fr1;

[m,n] = size(corner\_i);

for i = 1:20:n

Image\_corners = insertMarker(Image\_corners,[corner\_j(i) corner\_i(i)]);

end

figure();

imshow(Image\_corners);

hold on;

title('Corners (window size = 8, threshold = 20)', 'FontSize', 10);

1. **CornerOpticalFlow.m**

close all;

clear all;

%% Find Optical flow

%Read a video

v = VideoReader('video1.avi');

%Extract Frames from video

video = read(v);

%Frame No1

fr1 = video(:,:,:,1);

%Frame No1

fr2 = video(:,:,:,2);

subplot 211

imshow(fr1);

im1 = im2double(rgb2gray(fr1));

%im1 = imresize(im1t, 0.5); % downsize to half

subplot 212

imshow(fr2);

im2 = im2double(rgb2gray(fr2));

%im2 = imresize(im2t, 0.5); % downsize to half

windowSize = 8;

[u,v] = opticalFlow(im1,im2,windowSize);

%% Compute Corners

%eigenvalues threshold

thresh = 20;

windowSize = 8;

[corner\_i,corner\_j] = cornerDetection(im1,windowSize, thresh);

%% Compute optical flow for corners

[m,n] = size(corner\_i);

u\_deci = [];

v\_deci = [];

X\_deci = [];

Y\_deci = [];

% get coordinate for u and v in the original frame

for i = 1:n

X\_deci = [X\_deci corner\_j(i)];

Y\_deci = [Y\_deci corner\_i(i)];

u\_deci = [u\_deci u(corner\_i(i),corner\_j(i))];

v\_deci = [v\_deci v(corner\_i(i),corner\_j(i))];

end

figure();

imshow(fr1);

hold on;

% draw the velocity vectors

quiver(X\_deci, Y\_deci, u\_deci,v\_deci, 'y')

title('Optical flow on corners', 'FontSize', 10);

**4) VideoStabilization.m**

close all;

clear all;

%Read a video

v = VideoReader('video1.avi');

video = read(v);

[k,l,n,numFrames]= size(video);

%create the video object

video2 = VideoWriter('stabilized\_optical.avi');

open(video2); %open the file for writing

stabilized = video(:,:,:,1);

for i = 1:numFrames-1

fr1 = stabilized;

fr2 = video(:,:,:,i+1);

% downsize to half

im1t = im2double(rgb2gray(fr1));

im1 = imresize(im1t, 0.5); % downsize to half

im2t = im2double(rgb2gray(fr2));

im2 = imresize(im2t, 0.5); % downsize to half

windowSize = 8;

[u,v] = opticalFlow(im1,im2,windowSize);

stabilazed = WarpImage(im1,u,v);

im3 = imresize(stabilazed, 2); % upsize to double

writeVideo(video2,mat2gray(im3)); %write the image to file

end

close(video2); %close the file

implay('stabilized.avi');

**5) AffineStabilization.m**

close all;

clear all;

%Read a video

v = VideoReader('video1.avi');

video = read(v);

[k,l,n,numFrames]= size(video);

%create the video object

video2 = VideoWriter('stabilized\_affine.avi');

open(video2); %open the file for writing

stabilized = video(:,:,:,1);

for i = 1:numFrames-1

fr1 = stabilized;

fr2 = video(:,:,:,i+1);

% downsize to half

im1t = im2double(rgb2gray(fr1));

im1 = imresize(im1t, 0.5);

im2t = im2double(rgb2gray(fr2));

im2 = imresize(im2t, 0.5); % downsize to half

windowSize = 8;

[u,v] = opticalFlow(im1,im2,windowSize);

%eigenvalues threshold

thresh = 20;

windowSize = 8;

[corner\_i,corner\_j] = cornerDetection(im1,windowSize, thresh);

[m,n] = size(corner\_i);

%get matrices!!!

A = [];

b = [];

% get coordinate for u and v in the original frame

for i = 1:n

A2 = [corner\_j(i) corner\_i(i) 0 0 1 0 ; 0 0 corner\_j(i) corner\_i(i) 0 1];

b2 = [corner\_j(i) + u(corner\_i(i),corner\_j(i)); corner\_i(i) + v(corner\_i(i),corner\_j(i))];

A = [A ; A2];

b = [b ; b2];

end

t = pinv(A.'\*A)\*A'\*b;

T = [t(1) t(2) 0; t(3) t(4) 0; t(5) t(6) 1];

tform = affine2d(T);

stabilazed = imwarp(im1,tform);

im3 = imresize(stabilazed,[240,320]); % upsize to double

writeVideo(video2,mat2gray(im3)); %write the image to file

end

close(video2); %close the file