

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

**ΘΕΜΑΤΑ ΟΡΑΣΗΣ ΥΠΟΛΟΓΙΣΤΩΝ & ΓΡΑΦΙΚΗΣ**

**ΑΝΑΦΟΡΑ ΒΑΣΙΣΜΕΝΗ ΣΤΗΝ**

**2Η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ**

ΑΓΓΕΛΟΣ ΝΙΚΟΛΑΟΣ ΠΟΤΑΜΙΑΝΟΣ

Α.Μ. 1084537

up1084537@ac.upatras.gr

Πάτρα, 2024

**ΑΣΚΗΣΗ 1.**

imread

Διαβάζει εικόνες από αρχεία σε διάφορες μορφές (όπως JPEG, PNG, BMP) και τις φορτώνει ως πίνακες δεδομένων στο MATLAB. Είναι η βασική συνάρτηση για την εισαγωγή εικόνων στο περιβάλλον εργασίας.

imwarp

Εφαρμόζει γεωμετρικές μετασχηματίσεις σε εικόνες χρησιμοποιώντας αντικείμενα μετασχηματισμού (όπως affine2d ή projective2d). Επιτρέπει την εκτέλεση μετατοπίσεων, περιστροφών, κλιμακώνσεων και άλλων πολύπλοκων αλλαγών στην εικόνα.

affine2d

Δημιουργεί ένα αντικείμενο γεωμετρικής μετασχημάτισης affine 2D, το οποίο μπορεί να χρησιμοποιηθεί για την περιγραφή μετασχηματισμών όπως μετατοπίσεις, περιστροφές, κλιμακώσεις και συγχωνεύσεις. Οι affine μετασχηματισμοί διατηρούν τις ευθείες γραμμές και τις αναλογίες.

projective2d

Δημιουργεί ένα αντικείμενο γεωμετρικής μετασχημάτισης projective 2D, το οποίο επιτρέπει πιο πολύπλοκες μετασχηματίσεις που μπορούν να περιλαμβάνουν κλίσεις και αναλογικές αλλαγές, διατηρώντας τις γραμμές ευθείες αλλά όχι απαραίτητα τις παραλληλίες.

imref2d

Δημιουργεί ένα αντικείμενο αναφοράς χωρικών δεδομένων 2D, το οποίο ορίζει το σύστημα συντεταγμένων για μια εικόνα. Χρησιμοποιείται συχνά σε συνδυασμό με imwarp για τον καθορισμό της θέσης και του μεγέθους της μετασχηματισμένης εικόνας.

implay

Παίζει βίντεο ή ακολουθίες εικόνων στο MATLAB. Παρέχει ένα γραφικό περιβάλλον για την προβολή κινουμένων σχεδίων και μπορεί να χειριστεί διάφορες μορφές αρχείων βίντεο.

**ΑΣΚΗΣΗ 2**

Δημιουργείται ένας ευρύτερος καμβάς για την οριζόντια σύνθεση των κλιμακωμένων εικόνων. Αυτό επιτυγχάνεται με τον καθορισμό του πλάτους του καμβά ως τριπλάσιο του αρχικού πλάτους της εικόνας, εξασφαλίζοντας αρκετό χώρο για την τοποθέτηση πέντε διαφορετικών κλιμακώσεων:

canvas\_width = width \* 3; % Wide enough for 5 puddings  
canvas\_height = height; % Original height  
canvas = uint8(zeros(canvas\_height, canvas\_width, channels));

Οι παράγοντες κλιμάκωσης καθορίζονται γραμμικά χρησιμοποιώντας τη συνάρτηση linspace, η οποία παράγει πέντε τιμές από το 0.2 έως το 1.0, δημιουργώντας έτσι εικόνες σε αυξανόμενο μέγεθος:

scales = linspace(0.2, 1.0, 5);

Για να διασφαλιστεί η ομοιόμορφη κατανομή των κλιμακωμένων εικόνων στον καμβά, υπολογίζεται το συνολικό πλάτος που θα καταλαμβάνουν οι κλιμακώσεις καθώς και ο διαθέσιμος χώρος για τα διαστήματα μεταξύ τους:

total\_scaled\_width = 0;  
scaled\_widths = [];  
for i = 1:5  
 scaled\_w = round(width \* scales(i));  
 scaled\_widths(i) = scaled\_w;  
 total\_scaled\_width = total\_scaled\_width + scaled\_w;  
end  
  
total\_space = canvas\_width - total\_scaled\_width;  
space\_unit = total\_space / 16; % 4 gaps between 5 puddings

Οι θέσεις τοποθέτησης για κάθε κλιμακωμένη εικόνα υπολογίζονται λαμβάνοντας υπόψη το πλάτος και τον παράγοντα κλιμάκωσης, εξασφαλίζοντας ότι οι εικόνες τοποθετούνται με ομοιόμορφο διάστημα μεταξύ τους:

positions = zeros(1, 5);  
current\_pos = 1;  
**for** i = 1:5  
 positions(i) = current\_pos + scaled\_widths(i)/2;  
 **if** i < 5  
 current\_pos = current\_pos + scaled\_widths(i) + space\_unit \* (scales(i) + scales(i+1))/2;  
 **end**  
**end**

Για κάθε κλιμακωμένη εικόνα, δημιουργείται ένας μετασχηματισμός affine που εφαρμόζει την αντίστοιχη κλιμάκωση στην αρχική εικόνα. Η συνάρτηση imwarp χρησιμοποιείται για την εφαρμογή του μετασχηματισμού, ενώ η τοποθέτηση της κλιμακωμένης εικόνας στον καμβά γίνεται με προσοχή ώστε να αποφεύγεται η υπέρβαση των ορίων του καμβά:

transformations = {};  
**for** i = 1:5  
 transformations{i} = [scales(i), scales(i), positions(i), scaled\_widths(i)/2];  
end  
  
**for** i = 1:length(transformations)  
 scale\_x = transformations{i}(1);  
 scale\_y = transformations{i}(2);  
 pos\_x = transformations{i}(3);  
 pos\_y = transformations{i}(4);  
   
 tform = affine2d([scale\_x 0 0; 0 scale\_y 0; 0 0 1]);  
 ref = imref2d([round(height\*scale\_y), round(width\*scale\_x)]);  
 scaled\_img = imwarp(img, tform, 'OutputView', ref);  
   
 start\_y = max(1, round(pos\_y - (size(scaled\_img,1)/2)));  
 end\_y = min(canvas\_height, start\_y + size(scaled\_img,1) - 1);  
 start\_x = max(1, round(pos\_x - (size(scaled\_img,2)/2)));  
 end\_x = min(canvas\_width, start\_x + size(scaled\_img,2) - 1);  
   
 img\_start\_y = 1;  
 img\_start\_x = 1;  
 **if** start\_y < 1  
 img\_start\_y = 2 - start\_y;  
 start\_y = 1;  
 end  
 **if** start\_x < 1  
 img\_start\_x = 2 - start\_x;  
 start\_x = 1;  
 end  
   
 canvas(start\_y:end\_y, start\_x:end\_x, :) = scaled\_img(img\_start\_y:img\_start\_y+end\_y-start\_y, ...  
 img\_start\_x:img\_start\_x+end\_x-start\_x, :);  
end

A close up of food

Description automatically generated

**ΑΣΚΗΣΗ 3**

ορίζονται οι παράμετροι του βίντεο, συμπεριλαμβανομένου του αριθμού καρέ ανά δευτερόλεπτο (fps), της διάρκειας του βίντεο σε δευτερόλεπτα (duration), του συνολικού αριθμού καρέ (numFrames), καθώς και του μέγιστου ποσοστού στρέβλωσης (maxShear):

**fps** = 30; % Frames per second  
**duration** = 5; % Video duration in seconds  
**numFrames** = fps \* duration; % Total frames for 5 seconds  
**maxShear** = 0.5; % Maximum shear value

Για να διασφαλιστεί ότι η στρέβλωση δεν θα προκαλέσει παραμόρφωση ή αποκοπή της εικόνας, υπολογίζεται το επιπλέον πλάτος που απαιτείται στον καμβά ώστε να φιλοξενήσει τη μέγιστη στρέβλωση σε αμφότερες τις κατευθύνσεις:

extraWidth = round(height \* maxShear \* 2); % Account for maximum shear in both directions  
canvasWidth = width + extraWidth;  
outputView = imref2d([height, canvasWidth]);

Η τοποθέτηση της εικόνας σε έναν ευρύτερο καμβά επιτυγχάνεται με τον καθορισμό των ορίων του κόσμου (XWorldLimits και YWorldLimits), διατηρώντας την εικόνα κεντραρισμένη και τη βάση της σταθερή:

% Center the image in the wider canvas  
outputView.XWorldLimits = [-(extraWidth/2), width + (extraWidth/2)];  
outputView.YWorldLimits = [0, height];

Στη συνέχεια, δημιουργείται ένα αντικείμενο VideoWriter για την εγγραφή του βίντεο με το όνομα sheared\_pudding\_exercise3.avi, και καθορίζεται ο ρυθμός καρέ:

% **Create** VideoWriter **object**  
v = VideoWriter('sheared\_pudding\_exercise3.avi');  
v.FrameRate = fps;  
**open**(v);

Η κύρια διαδικασία δημιουργίας των καρέ με περιοδική στρέβλωση υλοποιείται μέσα σε έναν βρόχο for, ο οποίος επαναλαμβάνεται για κάθε καρέ του βίντεο. Σε κάθε επανάληψη, υπολογίζεται η τιμή της στρέβλωσης χρησιμοποιώντας τη συνάρτηση ημίτονου (sin), που εξασφαλίζει μια ομαλή και περιοδική κίνηση της στρέβλωσης γύρω από το κέντρο της εικόνας:

**for** frame = 1:numFrames  
 % Calculate shear **value** **using** sine **function** **for** smooth periodic motion  
 shearAmount = maxShear \* sin(2\*pi\*frame/(fps\*duration));  
   
 % **Create** shearing transformation matrix **for** horizontal shear  
 tform = affine2d([1 0 0; shearAmount 1 0; 0 0 1]);  
   
 % Apply shearing transformation **with** wider output **view**  
 shearedImg = imwarp(img, tform, 'OutputView', outputView);  
   
 % **Write** frame **to** video  
 writeVideo(v, shearedImg);  
**end**

Στον παραπάνω βρόχο, ο μετασχηματισμός affine δημιουργείται με τη μήτρα:

tform = affine2d([1 0 0; shearAmount 1 0; 0 0 1]);

Αυτή η μήτρα εφαρμόζει στρέβλωση μόνο κατά τον οριζόντιο άξονα, διατηρώντας ταυτόχρονα την κατακόρυφη διάσταση αμετάβλητη. Η συνάρτηση imwarp χρησιμοποιείται για την εφαρμογή του μετασχηματισμού στην αρχική εικόνα, με το αποτέλεσμα να τοποθετείται στον καθορισμένο καμβά (outputView):

**shearedImg** = imwarp(img, tform, 'OutputView', outputView);

Το επεξεργασμένο καρέ προστίθεται στο βίντεο μέσω της εντολής writeVideo:

writeVideo(v, shearedImg);

Μετά την ολοκλήρωση του βρόχου, το αρχείο βίντεο κλείνει με την εντολή:

% **Close** the video **file**  
**close**(v);

**ΑΣΚΗΣΗ 4**

ορίζονται οι παράμετροι του βίντεο, όπως το αριθμό των καρέ ανά δευτερόλεπτο (fps), η διάρκεια του βίντεο (duration), και ο συνολικός αριθμός καρέ (numFrames). Επιπλέον, καθορίζεται το μέγιστο ποσό στρέβλωσης (maxShear):

**fps** = 30; % Frames per second  
**duration** = 5; % Duration in seconds (increased to 5)  
**numFrames** = fps \* duration;  
**maxShear** = 0.5; % Maximum shear value

Για να διασφαλιστεί ότι η στρέβλωση δεν θα προκαλέσει υπέρβαση των ορίων της εικόνας, υπολογίζεται το επιπλέον πλάτος που απαιτείται στον καμβά:

extraWidth = round(width \* maxShear \* 2); % Double the extra width to accommodate both sides  
canvasWidth = width + extraWidth;

Στη συνέχεια, δημιουργείται ένα αντικείμενο VideoWriter για την εγγραφή του βίντεο με το όνομα 'sheared\_pudding.avi' και καθορίζεται το ρυθμό καρέ:

v = VideoWriter('sheared\_pudding.avi');  
v.FrameRate = fps;  
**open**(v);

Για την σωστή τοποθέτηση της εικόνας στον καμβά, δημιουργείται ένα αντικείμενο αναφοράς (imref2d) που καθορίζει τα όρια του κόσμου (XWorldLimits και YWorldLimits), διατηρώντας την βάση της εικόνας σταθερή:

outputView = imref2d([height, canvasWidth]);  
  
% Center the image in the canvas  
outputView.XWorldLimits = [-extraWidth/2, width + extraWidth/2];  
outputView.YWorldLimits = [0, height];

Η κύρια διαδικασία δημιουργίας των καρέ με περιοδική στρέβλωση πραγματοποιείται μέσα σε έναν βρόχο for, ο οποίος επαναλαμβάνεται για κάθε καρέ του βίντεο:

**for** frame = 1:numFrames  
 % Calculate shear **value** **using** sine **function** **for** smooth periodic motion  
 shearAmount = maxShear \* sin(2\*pi\*frame/numFrames); % Changed **to** sin **to** oscillate around center  
   
 % **Create** affine transformation matrix **for** horizontal shear  
 % **Add** translation **to** keep bottom fixed  
 tform = affine2d([1 0 0; shearAmount 1 0; -shearAmount\*height 0 1]);  
   
 % Apply transformation  
 shearedImg = imwarp(img, tform, 'OutputView', outputView);  
   
 % **Write** frame **to** video  
 writeVideo(v, shearedImg);  
**end**

Στο παραπάνω βρόχο, η τιμή της στρέβλωσης (shearAmount) υπολογίζεται χρησιμοποιώντας τη συνάρτηση ημίτονου (sin), που εξασφαλίζει μια ομαλή και περιοδική κίνηση της στρέβλωσης γύρω από το κέντρο της εικόνας. Ο μετασχηματισμός affine δημιουργείται με μήτρα που εφαρμόζει τη στρέβλωση κατά τον οριζόντιο άξονα, ενώ ταυτόχρονα προστίθεται μετατόπιση (translation) για να διατηρηθεί η βάση της εικόνας σταθερή:

tform = affine2d([1 0 0; shearAmount 1 0; -shearAmount\*height 0 1]);

Η συνάρτηση imwarp χρησιμοποιείται για την εφαρμογή του μετασχηματισμού στην εικόνα, με το αποτέλεσμα να τοποθετείται στον καθορισμένο καμβά (outputView):

**shearedImg** = imwarp(img, tform, 'OutputView', outputView);

**ΑΣΚΗΣΗ 5**

Η μάσκα μετατρέπεται σε ασπρόμαυρη και αντιστρέφεται για να εξασφαλιστεί η σωστή εφαρμογή της:

% Invert the mask  
mask = ~mask;

Στη συνέχεια, προσδιορίζεται το πλαίσιο (bounding box) της μάσκας ώστε να κοπεί το περιττό τμήμα της εικόνας:

% Find bounding box  
[row, col] = find(mask);  
crop\_top = min(row);  
crop\_bottom = max(row);  
crop\_left = min(col);  
crop\_right = max(col);

Οι εικόνες κοπέςται αναλόγως του bounding box:

% Crop images  
mask = mask(crop\_top:crop\_bottom, crop\_left:crop\_right);  
windmill = windmill(crop\_top:crop\_bottom, crop\_left:crop\_right, :);

Για να διευκολυνθεί η επεξεργασία και η δημιουργία του βίντεο, οι εικόνες κλιμακώνονται στο μισό του αρχικού τους μεγέθους:

% Resize  
**new**\_width = round(size(mask,2)/2);  
**new**\_height = round(size(mask,1)/2);  
mask = imresize(mask, [**new**\_height **new**\_width]);  
windmill = imresize(windmill, [**new**\_height **new**\_width]);  
background = imresize(background, 0.5);

Προσδιορίζονται οι διαστάσεις του φόντου και οι μετατοπίσεις ώστε ο ανεμόμυλλος να τοποθετείται κεντρικά στο φόντο:

% Get dimensions  
[**bg\_height, bg\_width,** ~] = size(**background);**y\_offset = round((**bg\_height** - new\_height)/2);  
x\_offset = round((**bg\_width** - new\_width)/2);

Δημιουργείται ένα αντικείμενο VideoWriter για την εγγραφή του βίντεο με το όνομα transf\_windmill\_matlab.avi και καθορίζεται ο ρυθμός καρέ:

% **Create** video writer  
v = VideoWriter('transf\_windmill\_matlab.avi');  
v.FrameRate = 30;  
**open**(v);

Ορίζονται οι παράμετροι της ανιμέισιον, όπως ο αριθμός των καρέ και το βήμα περιστροφής:

% Animation parameters  
num\_frames = 200;  
angle\_step = -360/num\_frames;

Η κύρια διαδικασία δημιουργίας των καρέ με περιστροφή των φτερωτών του ανεμόμυλου πραγματοποιείται μέσα σε έναν βρόχο for. Σε κάθε επανάληψη, υπολογίζεται η τρέχουσα γωνία περιστροφής, δημιουργείται η μήτρα περιστροφής, και εφαρμόζεται η περιστροφή στην εικόνα του ανεμόμυλου και στη μάσκα:

% Main loop  
**for** frame = 1:num\_frames  
 % Calculate rotation angle  
 angle = (frame-1) \* angle\_step;  
   
 % Create rotation matrix  
 R = [cosd(angle) -sind(angle); sind(angle) cosd(angle)];  
   
 % Rotate windmill and mask  
 rotated\_windmill = imrotate(windmill, angle, 'bilinear', 'crop');  
 rotated\_mask = imrotate(mask, angle, 'bilinear', 'crop');  
   
 % Create frame  
 current\_frame = background;  
   
 % Get ROI  
 roi = current\_frame(y\_offset+1:y\_offset+new\_height, x\_offset+1:x\_offset+new\_width, :);  
   
 % Apply mask  
 **for** c = 1:3  
 roi(:,:,c) = rotated\_windmill(:,:,c) .\* rotated\_mask + roi(:,:,c) .\* (1-rotated\_mask);  
 **end**  
   
 % Insert ROI back  
 current\_frame(y\_offset+1:y\_offset+new\_height, x\_offset+1:x\_offset+new\_width, :) = roi;  
   
 % Convert to uint8 before writing  
 current\_frame = im2uint8(current\_frame);  
   
 % Write frame  
 writeVideo(v, current\_frame);  
**end**

Στο παραπάνω βρόχο, η συνάρτηση imrotate χρησιμοποιείται για την περιστροφή τόσο της εικόνας του ανεμόμυλου όσο και της μάσκας με τη συγκεκριμένη γωνία. Η μάσκα εφαρμόζεται για να συνδυάσει φυσικά τον ανεμόμυλλο με το φόντο, διατηρώντας την οπτική συνέπεια:

rotated\_windmill = imrotate(windmill, angle, 'bilinear', 'crop');  
rotated\_mask = imrotate(mask, angle, 'bilinear', 'crop');

Η χρήση της μάσκας εξασφαλίζει ότι μόνο τα τμήματα του ανεμόμυλλου που αντιστοιχούν στη μάσκα θα εμφανίζονται πάνω στο φόντο, δημιουργώντας μια φυσική εμφάνιση:

% Apply mask  
**for** c = 1:3  
 roi(:,:,c) = rotated\_windmill(:,:,c) .\* rotated\_mask + roi(:,:,c) .\* (1-rotated\_mask);  
end

**ΑΣΚΗΣΗ 6**

Για την επανάληψη του προηγούμενου ερωτήματος με διαφορετικές μεθόδους παρεμβολής, δημιουργήθηκε μια συνάρτηση create\_windmill\_animation που δέχεται ως παράμετρο τη μέθοδο παρεμβολής (interp\_method). Η συνάρτηση αυτή εκτελεί τη διαδικασία δημιουργίας του βίντεο για κάθε συγκεκριμένη μέθοδο παρεμβολής:

% Function to create animation with different interpolation methods  
**function** **create\_windmill\_animation**(interp\_method)  
 **try**  
 % Read images  
 windmill = imread('windmill.png');  
 mask = imread('windmill\_mask.png');  
 background = imread('windmill\_back.jpeg');  
   
 % Convert images to double for processing  
 windmill = im2double(windmill);  
 mask = im2double(rgb2gray(mask));  
 background = im2double(background);  
   
 % Invert the mask  
 mask = ~mask;  
   
 % Find bounding box  
 [row,col] = find(mask);  
 crop\_top = min(row);  
 crop\_bottom = max(row);  
 crop\_left = min(col);  
 crop\_right = max(col);  
   
 % Crop images  
 mask = mask(crop\_top:crop\_bottom, crop\_left:crop\_right);  
 windmill = windmill(crop\_top:crop\_bottom, crop\_left:crop\_right, :);  
   
 % Resize  
 new\_width = round(size(mask,2)/2);  
 new\_height = round(size(mask,1)/2);  
 mask = imresize(mask, [new\_height new\_width]);  
 windmill = imresize(windmill, [new\_height new\_width]);  
 background = imresize(background, 0.5);  
   
 % Get dimensions  
 [bg\_height, bg\_width, ~] = size(background);  
 y\_offset = round((bg\_height - new\_height)/2);  
 x\_offset = round((bg\_width - new\_width)/2);  
   
 % Create video writer  
 filename = sprintf('transf\_windmill\_matlab\_%s.avi', interp\_method);  
 v = VideoWriter(filename);  
 v.FrameRate = 30;  
 open(v);  
   
 % Animation parameters  
 num\_frames = 200;  
 angle\_step = -360/num\_frames;  
   
 % Main loop  
 **for** frame = 1:num\_frames  
 % Calculate rotation angle  
 angle = (frame-1) \* angle\_step;  
   
 % Rotate windmill and mask using specified interpolation method  
 rotated\_windmill = imrotate(windmill, angle, interp\_method, 'crop');  
 rotated\_mask = imrotate(mask, angle, interp\_method, 'crop');  
   
 % Create frame  
 current\_frame = background;  
   
 % Get ROI  
 roi = current\_frame(y\_offset+1:y\_offset+new\_height, x\_offset+1:x\_offset+new\_width, :);  
   
 % Apply mask  
 **for** c = 1:3  
 roi(:,:,c) = rotated\_windmill(:,:,c) .\* rotated\_mask + roi(:,:,c) .\* (1-rotated\_mask);  
 **end**  
   
 % Insert ROI back  
 current\_frame(y\_offset+1:y\_offset+new\_height, x\_offset+1:x\_offset+new\_width, :) = roi;  
   
 % Convert to uint8 before writing  
 current\_frame = im2uint8(current\_frame);  
   
 % Write frame  
 writeVideo(v, current\_frame);  
 **end**  
 close(v);  
 **catch** e  
 fprintf('Error: %s\n', e.message);  
 **if** exist('v', 'var') && isopen(v)  
 close(v);  
 **end**  
 **end**  
**end**

Η συνάρτηση αυτή καλείται τρεις φορές με τις διαφορετικές μεθόδους παρεμβολής bilinear (αντίστοιχο με linear), nearest, και bicubic (αντίστοιχο με cubic):

% Run the animation **with** different interpolation methods  
% **In** MATLAB, the correct interpolation methods are:  
% 'bilinear' (instead **of** 'linear')  
% 'nearest'  
% 'bicubic' (instead **of** 'cubic')  
  
create\_windmill\_animation('bilinear');  
create\_windmill\_animation('nearest');  
create\_windmill\_animation('bicubic');

**Bilinear Interpolation (bilinear):** Η μέθοδος bilinear προσφέρει ομαλές μεταβάσεις χρωμάτων και μειώνει τις παραμορφώσεις κατά την περιστροφή των φτερωτών του ανεμόμυλλου. Το αποτέλεσμα είναι ένα βίντεο με υψηλή αισθητική ποιότητα, όπου οι φτερωτές κινούνται ομαλά χωρίς εμφανείς γραμμές ή σκαγγές.

**Nearest Neighbor Interpolation (nearest):** Η μέθοδος nearest παράγει πιο απλά και πιο «ψίχουλα» αποτελέσματα, με έντονες ακμές και περιορισμένη ομαλότητα στις περιστροφές. Αν και η μέθοδος αυτή είναι ταχύτερη, η ποιότητα του βίντεο είναι χαμηλότερη σε σύγκριση με τις άλλες μεθόδους, με εμφανείς αλλοιώσεις στα φτερωτά του ανεμόμυλλου.

**Bicubic Interpolation (bicubic):** Η μέθοδος bicubic προσφέρει την υψηλότερη ποιότητα μετασχηματισμού, με εξαιρετικά ομαλές περιστροφές και λεπτομερείς λεπτομέρειες στα φτερωτά. Το αποτέλεσμα είναι ένα βίντεο με εξαιρετική ποιότητα εικόνας, όπου οι περιστροφές φαίνονται φυσικές και χωρίς παραμορφώσεις.

**ΑΣΚΗΣΗ 7**

οι εικόνες κλιμακώνονται ώστε να ταιριάζουν στις διαστάσεις του φόντου. Η μπάλα και η μάσκα κλιμακώνονται σε συγκεκριμένο μέγεθος, ενώ το φόντο κλιμακώνεται στο μισό του αρχικού του μεγέθους:

% Resize images  
**background** = imresize(**background,** [720 1280]);  
**ball\_size** = 80;  
**ball** = imresize(**ball,** [**ball\_size ball\_size]);**mask = imresize(mask, [**ball\_size ball\_size]);**

Προσδιορίζονται οι διαστάσεις του φόντου και καθορίζονται οι αρχικές θέσεις της μπάλας στο πλαίσιο:

% Get dimensions  
[height, width, ~] = size(background);

Δημιουργείται ένα αντικείμενο VideoWriter για την εγγραφή του βίντεο με το όνομα transf\_beach\_ex7.avi και καθορίζεται ο ρυθμός καρέ:

% **Create** video writer  
v = VideoWriter('transf\_beach\_ex7.avi');  
v.FrameRate = 30;  
**open**(v);

Ορίζονται οι παράμετροι της ανιμέισιον, όπως ο αριθμός των καρέ (num\_frames), οι παράμετροι της κίνησης της μπάλας (βαρύτητα, ύψος αναπήδησης, οριζόντια ταχύτητα) και η αρχική θέση:

% Animation parameters  
num\_frames = 120;  
  
% Bouncing parameters  
gravity = 0.5;  
bounce\_height = 200;  
horizontal\_speed = 8;  
  
% Starting position  
x = width / 4;  
y = height - ball\_size - 50; % Start near bottom of frame  
  
% Initial vertical velocity (negative means going up)  
velocity = -20;  
  
% Rotation angle  
angle = 0;

Η κύρια διαδικασία δημιουργίας των καρέ με περιστροφή και αναπήδηση της μπάλας υλοποιείται μέσα σε έναν βρόχο for. Σε κάθε επανάληψη, ενημερώνεται η θέση της μπάλας με βάση την οριζόντια ταχύτητα και την κατακόρυφη ταχύτητα που επηρεάζεται από τη βαρύτητα. Όταν η μπάλα φτάσει στο έδαφος, η ταχύτητά της αναστρέφεται και μειώνεται για να προσομοιώσει την απώλεια ενέργειας κατά την αναπήδηση:

% Main animation loop  
**for** frame = 1:num\_frames  
 % Update position  
 x = x + horizontal\_speed;  
 velocity = velocity + gravity;  
 y = y + velocity;  
  
 % Bounce when hitting the ground  
 **if** y > height - ball\_size - 50 % Ground position  
 y = height - ball\_size - 50;  
 velocity = -abs(velocity \* 0.7); % Reduce bounce height by 30%  
 **end**  
  
 % Update rotation  
 angle = angle + 5; % Rotate 5 degrees per frame  
  
 % Rotate ball and mask using bilinear interpolation  
 rotated\_ball = imrotate(ball, angle, 'bilinear', 'crop');  
 rotated\_mask = imrotate(mask, angle, 'bilinear', 'crop');  
  
 % Create frame  
 current\_frame = background;  
  
 % Calculate ROI coordinates  
 y1 = max(1, round(y - ball\_size/2));  
 y2 = min(height, y1 + ball\_size - 1);  
 x1 = max(1, round(x - ball\_size/2));  
 x2 = min(width, x1 + ball\_size - 1);  
  
 % Make sure we don't exceed bounds  
 roi\_height = y2 - y1 + 1;  
 roi\_width = x2 - x1 + 1;  
   
 % Get corresponding regions from rotated images  
 ball\_roi = rotated\_ball(1:roi\_height, 1:roi\_width, :);  
 mask\_roi = rotated\_mask(1:roi\_height, 1:roi\_width);  
  
 % Get ROI from current frame  
 roi = current\_frame(y1:y2, x1:x2, :);  
  
 % Apply mask to each channel  
 **for** c = 1:3  
 roi(:,:,c) = ball\_roi(:,:,c) .\* mask\_roi + roi(:,:,c) .\* (1-mask\_roi);  
 **end**  
  
 % Insert ROI back into frame  
 current\_frame(y1:y2, x1:x2, :) = roi;  
  
 % Convert to uint8 before writing  
 current\_frame = im2uint8(current\_frame);  
  
 % Write frame  
 writeVideo(v, current\_frame);  
  
 % Reset ball position if it goes off screen  
 **if** x > width - ball\_size  
 x = ball\_size;  
 **end**  
**end**

Στο παραπάνω βρόχο, η συνάρτηση imrotate χρησιμοποιείται για την περιστροφή τόσο της εικόνας της μπάλας όσο και της μάσκας κατά 5 μοίρες ανά καρέ, εφαρμόζοντας την μέθοδο παρεμβολής 'bilinear' για ομαλή περιστροφή:

% Rotate ball and mask using bilinear interpolation  
rotated\_ball = imrotate(ball, angle, 'bilinear', 'crop');  
rotated\_mask = imrotate(mask, angle, 'bilinear', 'crop');

Η χρήση της μάσκας εξασφαλίζει ότι μόνο τα τμήματα της μπάλας που αντιστοιχούν στη μάσκα θα εμφανίζονται πάνω στο φόντο, δημιουργώντας μια φυσική εμφάνιση:

% Apply mask **to** each channel  
for **c** = 1:3  
 roi(:,:,**c**) = ball\_roi(:,:,**c**) .\* mask\_roi + roi(:,:,**c**) .\* (1-mask\_roi);  
end

**ΑΣΚΗΣΗ 8**

**Βήμα 1: Ανάγνωση και Προετοιμασία των Εικόνων**

Αρχικά, διαβάζονται οι απαραίτητες εικόνες: η εικόνα της μπάλας (ball.jpg), η μάσκα της μπάλας (ball\_mask.jpg), και το φόντο της παραλίας (beach.jpg). Οι εικόνες μετατρέπονται σε τύπο double για ακριβέστερη επεξεργασία.

% Read images  
ball = imread('ball.jpg');  
mask = imread('ball\_mask.jpg');  
background = imread('beach.jpg');  
  
% Convert images to double for processing  
ball = im2double(ball);  
mask = im2double(rgb2gray(mask));  
background = im2double(background);

**Βήμα 2: Αντιστροφή της Μάσκας**

Η μάσκα αντιστρέφεται για να διασφαλιστεί ότι οι περιοχές που θα εμφανιστεί η μπάλα θα αντιστοιχούν σωστά στην εικόνα του φόντου.

% Invert the mask  
mask = ~mask;

**Βήμα 3: Κλιμάκωση των Εικόνων**

Οι εικόνες κλιμακώνονται ώστε να ταιριάζουν στις διαστάσεις του φόντου. Η μπάλα και η μάσκα κλιμακώνονται σε συγκεκριμένο μέγεθος, ενώ το φόντο κλιμακώνεται στο μισό του αρχικού του μεγέθους για να διατηρηθεί η αναλογία διαστάσεων.

% Resize **background** to standard size  
**background** = imresize(**background,** [720 1280]);  
  
% Initial **ball** size **and** resize  
**ball\_size** = 100;  
**ball** = imresize(**ball,** [**ball\_size ball\_size]);**mask = imresize(mask, [**ball\_size ball\_size]);**

**Βήμα 4: Προσδιορισμός Διαστάσεων του Φόντου**

Καθορίζονται οι διαστάσεις του φόντου ώστε να υπολογιστούν οι θέσεις της μπάλας κατά τη διάρκεια της κίνησής της.

% Get dimensions  
[height, width, ~] = size(background);

**Βήμα 5: Δημιουργία Αντικειμένου VideoWriter**

Δημιουργείται ένα αντικείμενο VideoWriter για την εγγραφή του βίντεο με το όνομα transf\_beach.avi και καθορίζεται ο ρυθμός καρέ σε 30 fps.

% **Create** video writer  
v = VideoWriter('transf\_beach.avi');  
v.FrameRate = 30;  
**open**(v);

**Βήμα 6: Ορισμός Παραμέτρων Ανιμέισιον**

Ορίζονται οι παράμετροι της ανιμέισιον, όπως ο αριθμός των καρέ (num\_frames), οι παράμετροι της κίνησης της μπάλας (βαρύτητα, ύψος αναπήδησης, οριζόντια ταχύτητα), και η αρχική θέση της μπάλας.

% Animation parameters  
num\_frames = 150;  
  
% Create time vector  
t = linspace(0, 1, num\_frames);  
  
% Starting positions  
start\_x = width / 4;  
start\_y = height / 2;  
  
% Calculate trajectories  
x\_positions = linspace(start\_x, width \* 3/4, num\_frames);  
y\_positions = start\_y - 200 \* sin(pi \* t);  
  
% Scale factors (ball gets smaller as it moves)  
scale\_factors = linspace(1.0, 0.3, num\_frames);  
  
% Rotation angles  
rotation\_angles = linspace(0, 360, num\_frames);

Σε αυτή τη φάση, οι θέσεις της μπάλας στον οριζόντιο άξονα (x\_positions) και στον κατακόρυφο άξονα (y\_positions) καθορίζονται έτσι ώστε η μπάλα να κινείται προς τη θάλασσα και να εκφυλίζεται στον ορίζοντα. Η ταχύτητα και η κλίμακα της μπάλας αλλάζουν δυναμικά κατά τη διάρκεια της κίνησης της.

**Βήμα 7: Κύρια Διαδικασία Δημιουργίας Καρέ**

Μέσα σε έναν βρόχο for, για κάθε καρέ, υπολογίζεται η τρέχουσα θέση, η κλίμακα και η γωνία περιστροφής της μπάλας. Η μπάλα περιστρέφεται και κλιμακώνεται αναλόγως, ενώ το καρέ συντίθεται με το φόντο χρησιμοποιώντας τη μάσκα για ομαλή ενσωμάτωση.

% Main animation **loop**  
**for** frame = 1:num\_frames  
 % **Get** **current** position **and** parameters  
 current\_x = round(x\_positions(frame));  
 current\_y = round(y\_positions(frame));  
 scale = scale\_factors(frame);  
 angle = rotation\_angles(frame);  
  
 % Scale ball **and** mask  
 current\_size = max(1, round(ball\_size \* scale));  
 scaled\_ball = imresize(ball, [current\_size current\_size]);  
 scaled\_mask = imresize(mask, [current\_size current\_size]);  
  
 % Rotate ball **and** mask **using** bilinear interpolation  
 rotated\_ball = imrotate(scaled\_ball, angle, 'bilinear', 'crop');  
 rotated\_mask = imrotate(scaled\_mask, angle, 'bilinear', 'crop');  
  
 % **Create** frame  
 current\_frame = background;  
  
 % Calculate ROI coordinates  
 y1 = max(1, round(current\_y - current\_size/2));  
 y2 = min(height, y1 + current\_size - 1);  
 x1 = max(1, round(current\_x - current\_size/2));  
 x2 = min(width, x1 + current\_size - 1);  
  
 % Calculate **valid** regions  
 roi\_height = y2 - y1 + 1;  
 roi\_width = x2 - x1 + 1;  
  
 % **Get** regions **from** rotated images  
 ball\_roi = rotated\_ball(1:roi\_height, 1:roi\_width, :);  
 mask\_roi = rotated\_mask(1:roi\_height, 1:roi\_width);  
  
 % **Get** ROI **from** **current** frame  
 roi = current\_frame(y1:y2, x1:x2, :);  
  
 % Apply mask **to** **each** channel  
 **for** c = 1:3  
 roi(:,:,c) = ball\_roi(:,:,c) .\* mask\_roi + roi(:,:,c) .\* (1 - mask\_roi);  
 **end**  
  
 % **Insert** ROI back **into** frame  
 current\_frame(y1:y2, x1:x2, :) = roi;  
  
 % Convert **to** uint8 **before** writing  
 current\_frame = im2uint8(current\_frame);  
  
 % **Write** frame  
 writeVideo(v, current\_frame);  
**end**

Σε αυτό το βρόχο:

Θέση και Παράμετροι: Για κάθε καρέ, η τρέχουσα θέση της μπάλας (current\_x, current\_y), η κλίμακα (scale), και η γωνία περιστροφής (angle) υπολογίζονται βάσει των προεπιλεγμένων ακολουθιών.

Κλιμάκωση και Περιστροφή: Η μπάλα και η μάσκα κλιμακώνονται ανάλογα με την τρέχουσα κλίμακα και περιστρέφονται κατά την αντίστοιχη γωνία χρησιμοποιώντας την μέθοδο παρεμβολής 'bilinear' για ομαλές περιστροφές.

Σύνθεση Καρέ: Η περιστραμμένη και κλιμακωμένη μπάλα συντίθεται με το φόντο χρησιμοποιώντας τη μάσκα, εξασφαλίζοντας ότι η μπάλα ενσωματώνεται φυσικά στο σκηνικό.

Εγγραφή Καρέ: Το τελικό καρέ μετατρέπεται σε τύπο uint8 και εγγράφεται στο βίντεο.

**Βήμα 8: Κλείσιμο του Αρχείου Βίντεο**

Μετά την ολοκλήρωση του βρόχου, το αρχείο βίντεο κλείνει και εμφανίζεται ένα μήνυμα επιβεβαίωσης.

% **Close** video writer  
**close(v);  
disp('Beach ball** horizon animation created successfully!');

**Βήμα 9: Εκτέλεση της Συνάρτησης**

Τέλος, καλείται η συνάρτηση για να ξεκινήσει η διαδικασία δημιουργίας του βίντεο.

% Run the **function**  
**create\_beach\_ball\_horizon\_animation**();