Πολυτεχνική Σχολή
Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Θέματα Όρασης Υπολογιστών

ΕΡΓΑΣΤΗΡΙΑΚΉ ΆΣΚΗΣΗ 2

Κατσαρός Ανδρέας
Α.Μ. 1084522

Πάτρα, 2024-25

Ερωτήματα:

Μπορείτε να δείτε όλα τα video/results στο φακελο after_implementation_results.Στην αναφορά υπάρχουν ενδεικτικά screenshots.

Ερώτημα 1:

Σύντομη Περιγραφή Συναρτήσεων MATLAB για Γεωμετρικούς Μετασχηματισμούς και Επεξεργασία Εικόνας

1. **imread**
   - **Περιγραφή:** Διαβάζει μια εικόνα από αρχείο και επιστρέφει έναν πίνακα που αναπαριστά τα δεδομένα της εικόνας στο MATLAB.
   - **Χρήση:**
     img = imread('image.jpg');
   - **Εφαρμογή:** Χρησιμοποιείται για την εισαγωγή εικόνων στο MATLAB για επεξεργασία.
2. **imwarp**
   - **Περιγραφή:** Εφαρμόζει γεωμετρικό μετασχηματισμό (π.χ., affine, projective) σε μια εικόνα.
   - **Χρήση:**
     transformedImg = imwarp(img, tform); όπου tform είναι ένα αντικείμενο μετασχηματισμού (π.χ., affine2d, projective2d).
   - **Εφαρμογή:** Χρησιμοποιείται για τη μετατόπιση, περιστροφή, κλίμακα, ή παραμόρφωση εικόνων.
3. **affine2d**
   - **Περιγραφή:** Δημιουργεί ένα αντικείμενο μετασχηματισμού 2D affine, το οποίο ορίζει μετατοπίσεις, περιστροφές, κλίμακα, και παραμορφώσεις χωρίς προοπτική.
   - **Χρήση:**
     tform = affine2d([a b 0; c d 0; e f 1]); όπου οι συντελεστές ορίζουν τον μετασχηματισμό.
   - **Εφαρμογή:** Χρήσιμο για γεωμετρικές τροποποιήσεις εικόνας.
4. **projective2d**
   - **"Περιγραφή:** Δημιουργεί ένα αντικείμενο μετασχηματισμού **2D projective**, που επιτρέπει **γεωμετρικές παραμορφώσεις**."
   - **"Χρήση:** tform = projective2d([a b c; d e f; g h i]); όπου οι συντελεστές ορίζουν έναν γενικό μετασχηματισμό με **οπτικές ιδιότητες μετασχηματισμού**."
   - **"Εφαρμογή:** Χρησιμοποιείται σε πιο σύνθετες εφαρμογές, όπως η ανακατασκευή εικόνας ή η διόρθωση προοπτικής."
5. **imref2d**
   - **Περιγραφή:** Ορίζει τις γεωμετρικές αναφορές για μια 2D εικόνα, όπως διαστάσεις και κλίμακα, βοηθώντας σε ακριβείς μετασχηματισμούς.
   - **Χρήση:**
   - ref = imref2d(size(img), xWorldLimits, yWorldLimits);
     Εφαρμογή: Χρήσιμο για την ευθυγράμμιση εικόνων στον φυσικό χώρο.
6. **implay**
   - **Περιγραφή:** Αναπαράγει σειρές εικόνων ή βίντεο σε περιβάλλον γραφικού αναπαραγωγέα.
   - **Χρήση:** implay(videoData);
     Εφαρμογή: Εξετάζει κινούμενες εικόνες ή βίντεο καρέ-καρέ για ανάλυση ή παρουσίαση.

Οι παραπάνω συναρτήσεις παρέχουν τα βασικά εργαλεία για την επεξεργασία και τον γεωμετρικό μετασχηματισμό εικόνων στο MATLAB, επιτρέποντας ευέλικτες εφαρμογές στον τομέα της υπολογιστικής όρασης και της επεξεργασίας εικόνας.
Στην υπόλοιπη άσκηση πολλές από αυτές χρησιμοποιούνται για τον κώδικα λύσης.

Ερώτηση 2:

Ανάλυση Λογικής και Κώδικα για την Κατασκευή Σύνθετης Εικόνας με Κλιμακώσεις
Η εκφώνηση αφορά τη δημιουργία μιας σύνθετης εικόνας που περιλαμβάνει πολλαπλές κλιμακώσεις μιας αρχικής εικόνας, μέσω γεωμετρικών μετασχηματισμών και τοποθέτησης σε καμβά.
Ο κώδικας έχει στο τέλος και αρκετά σχόλια με στόχο την πλήρη κατανόηση του.

Λογική και Διαδικασία Κώδικα

1. **Φόρτωση Εικόνας και Καμβάς:**
   - Διαβάζεται η αρχική εικόνα (input_image = imread('pudding.png').
   - Δημιουργείται ένας καμβάς μεγέθους 500×1500 (pixels) με μαύρο φόντο, στον οποίο θα τοποθετηθούν οι κλιμακωμένες εικόνες. canvas = uint8(zeros(500, 1500, 3));

```
% Load the image
input_image = imread('pudding.png');

% Create a blank canvas (black background)
canvas = uint8(zeros(500, 1500, 3)); % Canvas dimensions: 500x1500
```

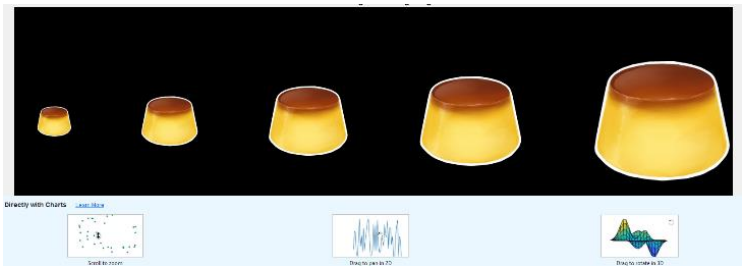
2. **Ορισμός Παραμέτρων:**
   - Παράγοντες κλιμάκωσης S={0.5,0.75,1,1.25,1.5} καθορίζουν την αλλαγή μεγέθους.
   - Αρχική θέση στον καμβά: x=10
   - Απόσταση μεταξύ εικόνων: d=20.

```
% Define scaling factors and parameters
scaling_factors = [0.3, 0.5, 0.7, 0.9, 1.2];
start_x = 50; % Initial X position
spacing = 150; % Space between scaled images
```

3. **Κλιμάκωση και Τοποθέτηση:**
   - Για κάθε παράγοντα s∈Ss \in S:
     - **Κλιμάκωση:** Η εικόνα κλιμακώνεται με τον παράγοντα s:
       $$x′=s⋅x, y′=s⋅y$$
     - **Υπολογισμός Θέσης:** Η εικόνα τοποθετείται οριζόντια στον καμβά, κεντραρισμένη κάθετα.
     - **Ενημέρωση Θέσης:** Η επόμενη κλιμακωμένη εικόνα τοποθετείται
       $$x' = x + w + d$$
       όπου w είναι το πλάτος της κλιμακωμένης εικόνας.
       Κώδικας:

```
scaled_image = imresize(input_image, scale);
[rows, cols, ~] = size(scaled_image);
y_position = round((size(canvas, 1) - rows) / 2);
canvas(y_position:y_position+rows-1, current_x:current_x+cols-  1,:)=scaled_image;
                                current_x = current_x + cols + spacing;
```

4. **Εμφάνιση και Αποθήκευση:**
   - Το τελικό αποτέλεσμα εμφανίζεται και αποθηκεύεται ως linear_scaled_puddings.png.
   - Για κάθε εικόνα κλιμάκωσης , έχουμε ένα μετασχηματισμό affine που εφαρμόζει την αντίστοιχη κλιμάκωση στην αρχική εικόνα. Η συνάρτηση imwarp εφαρμόζεται για τον μετασχηματισμό, ενώ η τοποθέτηση της κλιμακωμένης εικόνας στον καμβά γίνεται με προσοχή ώστε να αποφεύγεται η υπέρβαση των ορίων του καμβά:



Μαθηματική Θεμελίωση

- **Κλιμάκωση Εικόνας:** Η κλιμάκωση ορίζεται από τον πίνακα μετασχηματισμού:

$$S=|s\ 0|$$
$$|0\ s|$$

και εφαρμόζεται σε κάθε σημείο (x,y)(x, y) της εικόνας ως:

$$x'=S\cdot x$$

όπου ss είναι ο παράγοντας κλίμακας.

- **Τοποθέτηση στον Καμβά:** Η θέση υπολογίζεται ως:

$$xcurrent=xtcurrent+w+\ d$$

και το κάθετο κεντράρισμα δίνεται από:

$$ynew=(Hcanvas-h)/2$$

όπου Ηκαμβάς είναι το ύψος του καμβά.

Αποτέλεσμα

Το πρόγραμμα παράγει μια σύνθεση εικόνων διαφορετικών κλιμακώσεων, τοποθετημένων ομοιόμορφα στον καμβά. Η τελική εικόνα δείχνει πώς μεταβάλλεται η αρχική εικόνα για διάφορους παράγοντες κλίμακας, αναδεικνύοντας τη χρήση γεωμετρικών μετασχηματισμών.

Ερώτημα 3:

Ανάλυση Λογικής και Κώδικα για Δημιουργία Περιοδικής Ακολουθίας Εικόνων με Στρέβλωση
Η άσκηση αποσκοπεί στη δημιουργία μιας περιοδικής ακολουθίας εικόνων (βίντεο) από την εικόνα pudding.png, μέσω μετασχηματισμών στρέβλωσης (shearing) χρησιμοποιώντας MATLAB.

Λογική του Κώδικα και Μαθηματική Προσέγγιση

1. Φόρτωση εικόνας και αρχικοποίηση

Η εικόνα pudding.png φορτώνεται στη μνήμη. Ορίζεται ένας αριθμός καρέ (frames) για την ακολουθία, ενώ δεν χρειάζεται καμβάς καθώς κάθε καρέ επεξεργάζεται ξεχωριστά.

```
% Get image dimensions
[rows, cols, ~] = size(img);

% Create reference object for the image
ref = imref2d(size(img));

% Number of frames for the animation
numFrames = 60;

% Initialize cell array to store transformed images
transformedImages = cell(1, numFrames);
```

2. Περιοδική στρέβλωση :Χρησιμοποιήθηκε η shear.

Ορίζεται h περιοδική συνάρτηση createComplexShearing(t, amplitude) που καθορίζει τη στρέβλωση της εικόνας σε κάθε καρέ. Η στρέβλωση δίνεται από τις παραμέτρους shearX και shearY:

$$shearX = A \cdot (\sin(2\pi t) + 0.5\sin(4\pi t))$$
$$shearY = A \cdot (\cos(2\pi t) + 0.5\cos(4\pi t))$$

όπου A είναι το amplitude της στρέβλωσης και t είναι ο χρόνος μεταξύ 0 και 1.

```
% Function to create a more complex shearing pattern (optional)
function tform = createComplexShearing(t, amplitude)

    % Combine multiple sinusoidal components for more interesting motion
    shearX = amplitude * (sin(2*pi*t) + 0.5*sin(4*pi*t));
    shearY = amplitude * (cos(2*pi*t) + 0.5*cos(4*pi*t));
```

3. Εφαρμογή μετασχηματισμού

Για κάθε καρέ υπολογίζεται ο affine μετασχηματισμός με βάση τον πίνακα και τους μετασχηματισμους shearing:

$$T = \begin{vmatrix} 1 & shearX & 0 \\ shearY & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Αυτός ο πίνακας στρέβλωνει την εικόνα οριζόντια και κατακόρυφα ανάλογα με τις τιμές shearX και shearY.

Η συνάρτηση imwarp εφαρμόζει τον παραπάνω μετασχηματισμό στην εικόνα για κάθε τιμή του t.

```
% Create transformation matrix
 tform = affine2d([1    shearX  0;
                   shearY   1    0;
                   0      0     1]);
```

4. Σύνθεση βίντεο

Οι μετασχηματισμένες εικόνες αποθηκεύονται σε έναν πίνακα. Το MATLAB χρησιμοποιεί τη συνάρτηση implay για την αναπαραγωγή του βίντεο, ενώ το αποτέλεσμα αποθηκεύεται σε αρχείο AVI για μικρό μέγεθος.

Μαθηματική Ανάλυση

- Affine shear matrix: Ο affine πίνακας στρέβλωσης εφαρμόζει γεωμετρική παραμόρφωση στην εικόνα. Το αποτέλεσμα για κάθε συντεταγμένη (x, y) δίνεται από τη σχέση:

$$x' = x + shearX \cdot y$$
$$y' = shearY \cdot x + y$$

- Περιοδική στρέβλωση: Οι ημιτονοειδείς συναρτήσεις εξασφαλίζουν ότι οι τιμές shearX και shearY επαναλαμβάνονται περιοδικά, δημιουργώντας ομαλή κινούμενη εικόνα.

Επεξήγηση Κώδικα

- Ο βρόχος υπολογίζει τις παραμέτρους shearX και shearY για κάθε καρέ, σύμφωνα με τις περιοδικές συναρτήσεις.
- Ο affine πίνακας χρησιμοποιείται από τη συνάρτηση imwarp για την εφαρμογή του μετασχηματισμού στην εικόνα.
- Το αποτέλεσμα αποθηκεύεται ως βίντεο AVI, ενώ παράλληλα μπορεί να προβληθεί με τη συνάρτηση που εξηγήθηκε implay. Έχει υλοποιηθεί επιτυχώς δείχνωντας μια περιοδική α κολουθία εικόνων (βίντεο) με την χρήση shearing.
- ONOMA VIDEO: pudding_shearing.avi

Συμπεράσματα

Η άσκηση εφαρμόζει περιοδικούς affine μετασχηματισμούς στρέβλωσης για τη δημιουργία ενός βίντεο όπου η εικόνα παραμορφώνεται περιοδικά. Το αποτέλεσμα δείχνει τη χρήση γεωμετρικών μετασχηματισμών και περιοδικών συναρτήσεων στην παραγωγή κινούμενων εικόνων.

Ερώτημα 4:

Το αποτέλεσμα θα αποθηκευτεί σε αρχείο "sheared_pudding.avi".

Λογική και Μαθηματική Προσέγγιση

- **Στρέβλωση κατά τον οριζόντιο άξονα:**
  Χρησιμοποιούμε affine μετασχηματισμό με στρέβλωση μόνο στον οριζόντιο άξονα (shearX), διατηρώντας τον κατακόρυφο άξονα αμετάβλητο.
- **Περιοδικότητα:**
  Η τιμή της οριζόντιας στρέβλωσης υπολογίζεται με μια ημιτονοειδή συνάρτηση:

$$shearX = maxShear \cdot \sin(2\pi t)$$

όπου t μεταβάλλεται με την πάροδο του χρόνου από 0 έως 1. Αυτή η επιλογή εξασφαλίζει ομαλές, επαναλαμβανόμενες μεταβολές (περιοδικότητα) στη στρέβλωση.

- **Affine Μετασχηματισμός:**
  Το μετασχηματισμένο μητρώο για μόνο οριζόντια στρέβλωση είναι:
  - $T = \begin{vmatrix} 1 & shearX & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$

```
% Initialize cell array to store transformed images
transformedImages = cell(1, numFrames);

% Parameters for periodic shearing
maxShear = 0.5;  % Maximum shearing amount
period = 2*pi;   % Full period of the sine wave

% Create periodic horizontal shearing animation with fixed base
for i = 1:numFrames
   % Calculate shearing factor using sine function for smooth periodicity
   % Only horizontal shearing (shearX), no vertical shearing
   t = (i-1)/numFrames;  % Normalized time from 0 to 1
   shearX = maxShear * sin(period * t);
```

Αυτό το μητρώο εφαρμόζεται στην εικόνα για να παραμορφώσει οριζόντια, διατηρώντας το βασικό σχήμα σταθερό.

- **Υλοποίηση στον Κώδικα:**
  <u>Αντίστοιχη με το παραπάνω ερώτημα</u>
  - Το βρόχο for διατρέχει τους χρονικούς δείκτες για numFrames καρέ.
  - Σε κάθε επανάληψη, υπολογίζεται η τιμή της στρέβλωσης με βάση την ημιτονοειδή συνάρτηση.
  - Η συνάρτηση imwarp εφαρμόζει τον affine μετασχηματισμό σε κάθε καρέ.
  - Τα παραγόμενα καρέ συναρμολογούνται σε βίντεο, το οποίο αποθηκεύεται ως "sheared_pudding.avi".

```
% Only horizontal shearing (shearX), no vertical shearing
   t = (i-1)/numFrames;  % Normalized time from 0 to 1
   shearX = maxShear * sin(period * t);

   % Create affine transformation matrix for horizontal-only shearing
   % The matrix is designed to keep the base fixed by adjusting the translation
   tform = affine2d([1    shearX  0;
                     0    1       0;
                     0    0       1]);

   % Apply the transformation
   transformedImages{i} = imwarp(img, tform, 'OutputView', ref);
```

Έτσι συμπεραίνουμε ότι μπορούμε:

- **Ομαλή Περιοδική Στρέβλωση:**
  Να παρατηρήσουμε πώς η εικόνα "pudding.png" στρέβλωται περιοδικά μόνο οριζόντια, χωρίς να αλλοιώνεται η βασική της δομή.

- **Περιοδικότητα με Σταθερή Βάση:**
  Η χρήση της ημιτονοειδούς συνάρτησης καθιστά τη στρέβλωση επαναλαμβανόμενη και ομαλή, ενώ η σταθερή βάση διατηρεί το κέντρο της εικόνας απαλλαγμένο από μετατοπίσεις, προσδίδοντας συνοχή στο animation.

- **Απλότητα και Αποτελεσματικότητα του Μετασχηματισμού:**
  Μέσα από τον κώδικα, επιδεικνύεται πως απλές μαθηματικές συναρτήσεις και affine μετασχηματισμοί μπορούν να δημιουργήσουν περίπλοκες κινούμενες εικόνες με περιοδικότητα, δίνοντας πρακτική επίδειξη των θεωρητικών εννοιών της γεωμετρικής επεξεργασίας εικόνας.

Με αυτόν τον τρόπο, επιτυγχάνουμε μια ομαλή, περιοδική οριζόντια στρέβλωση της εικόνας, καταδεικνύοντας πρακτικά τα μαθηματικά εργαλεία (ημιτονοειδή συναρτήσεις και affine μετασχηματισμούς) για τη δημιουργία κινούμενης εικόνας.
Ενδεικτικά


Ερώτημα 5:

Ζητείται η δημιουργία ενός βίντεο, χρησιμοποιώντας περιστροφή, κλίμακα, μετατόπιση και ενσωμάτωση μάσκας για φυσική εμφάνιση του ανεμόμυλου.

Βασικά Βήματα του Κώδικα

1. Φόρτωση και Προεπεξεργασία Εικόνων:
   - Φορτώνονται οι εικόνες: windmill.png, μάσκα windmill_mask.png και φόντο windmill_back.jpeg.
   - Μετατροπή των εικόνων σε floating-point μορφή (im2double) και προετοιμασία της μάσκας (αντιστροφή, ώστε τα φτερά να είναι τα “άσπρα” μέρη).

```
% Load images
rotor = imread('windmill.png');          % Windmill blades
mask = imread('windmill_mask.png');      % Mask for blades
bg = imread('windmill_back.jpeg');       % Background image

% Convert to double precision
rotor = im2double(rotor);
mask = im2double(rgb2gray(mask));
bg = im2double(bg);
```

2. Cutting με Μάσκα:
   - Εντοπίζεται το περιγράψιμο πλαίσιο (bounding box) της μάσκας για να απομονωθούν τα winds.
   - Κόβονται η μάσκα και η εικόνα των φτερών στο επιλεγμένο πλαίσιο.
   - Οι απομονωμένες περιοχές αναπροσαρμόζονται (resize) για αποτελεσματικότερη επεξεργασία και σύμπτωση με το φόντο.

3. Προσδιορισμός Τοποθεσίας στο Φόντο:
    - Οι διαστάσεις του φόντου και οι μετατοπίσεις (offsets) υπολογίζονται ώστε οι περιστρεφόμενες φτερούγες να τοποθετούνται σωστά πάνω στο φόντο.
    - Για κάθε καρέ, αφού περιστραφούν οι φτερούγες και η μάσκα, υπολογίζεται το νέο καρέ Ik ως εξής:

$$Ik(x,y)=mask\_rot(x,y)\cdot rotor\_rot(x,y)+[1-mask\_rot(x,y)]\cdot background(x,y)$$

```
% Find bounding box of the mask
[r, c] = find(mask);
top = min(r);
bottom = max(r);
left = min(c);
right = max(c);

% Crop windmill and mask
mask_c = mask(top:bottom, left:right);
rotor_c = rotor(top:bottom, left:right, :);

% Resize for efficiency
new_w = round(size(mask_c, 2) / 2);
new_h = round(size(mask_c, 1) / 2);
mask_r = imresize(mask_c, [new_h, new_w]);
rotor_r = imresize(rotor_c, [new_h, new_w]);
bg_r = imresize(bg, 0.5);

% Background dimensions and offsets
[bg_h, bg_w, ~] = size(bg_r);
y_off = round((bg_h - new_h) / 2);
x_off = round((bg_w - new_w) / 2);
```

4. Αρχικοποίηση Βίντεο:
    - Ορίζεται ένας VideoWriter για να αποθηκευτεί το αποτέλεσμα ως αρχείο (π.χ., windmill_rot.avi) με καθορισμένο FrameRate.

5. Βρόχος Animation:
    - Περιστροφή: Για κάθε καρέ, υπολογίζεται η τρέχουσα γωνία περιστροφής:

$$angle = (k - 1) \times a\_step$$

όπου a_step = -360 / n_frames καθορίζει τη σταθερή μεταβολή της γωνίας. .

    - Οι συναρτήσεις imrotate και affine πίνακες υλοποιούν αυτή την πράξη.

    - Εφαρμογή Μετασχηματισμού:
        - Χρησιμοποιείται η συνάρτηση imrotate για περιστροφή των φτερών και της μάσκας κατά τη γωνία angle.
        - Μαθηματικά, η περιστροφή εφαρμόζεται μέσω ενός affine μετασχηματισμού που υπολογίζει τις νέες συντεταγμένες κάθε pixel.
    - Ενσωμάτωση στο Φόντο:
        - Το περιστρεφόμενο μέρος (φτερούγες) τοποθετείται πάνω στο φόντο στη θέση που έχει υπολογιστεί.
        - Χρησιμοποιείται η μάσκα για να γίνει blending:

$$pixel = rotor\_rot \times mask\_rot + background \times (1 - mask\_rot)$$

που εξασφαλίζει φυσικό συνδυασμό χωρίς άκρες.
    - Αποθήκευση Καρέ: Το κάθε καρέ γράφεται στο βίντεο.

```
n_frames = 200;              % Total number of frames
a_step = -360 / n_frames;    % Rotation step per frame

for k = 1:n_frames
    % Current rotation angle
    angle = (k - 1) * a_step;

    % Rotate windmill and mask
    rotor_rot = imrotate(rotor_r, angle, 'bilinear', 'crop');
    mask_rot = imrotate(mask_r, angle, 'nearest', 'crop');

    % Overlay on background
    frame = bg_r; % Start with resized background
    roi = frame(y_off+1:y_off+new_h, x_off+1:x_off+new_w, :);

    % Blend using mask
    for ch = 1:3
        roi(:,:,ch) = rotor_rot(:,:,ch) .* mask_rot + roi(:,:,ch) .* (1 - mask_rot);
    end

    % Insert ROI back into the frame
    frame(y_off+1:y_off+new_h, x_off+1:x_off+new_w, :) = roi;

    % Write frame to video
    writeVideo(vid, im2uint8(frame));

    % Optional display for debugging
    imshow(frame);
    title(sprintf('Frame %d of %d', k, n_frames));
    pause(0.01);
end
```

6. Ολοκλήρωση:
    - Ο βρόχος ολοκληρώνεται μετά από 200 καρέ, το βίντεο κλείνει και αποθηκεύεται ως windmill_rot.avi.

```
writeVideo(vid, im2uint8(frame));
```

Λογική και Συμπεράσματα

- **Κατανόηση Μετασχηματισμών:** Καταφέρνουμε να δείξουμε πως εφαρμόζονται μαθηματικοί affine μετασχηματισμοί (όπως η περιστροφή) σε εικόνες.
- **Περιοδική Κίνηση:** Η χρήση μιας σταθερής, περιοδικής μεταβολής της γωνίας περιστροφής εξασφαλίζει ομαλή και συνεχόμενη κίνηση στις φτερούγες. Η ρύθμιση των offsets διασφαλίζει ότι η βάση του ανεμοπτέρου παραμένει σταθερή στο φόντο
- **Χρήση Μάσκας για Φυσική Ενσωμάτωση:** Η ενσωμάτωση της μάσκας δείχνει πώς μπορεί κάποιος να συνδυάσει δύο εικόνες (εδώ τα φτερά και το φόντο) με φυσικό τρόπο, χωρίς να εμφανίζονται άσχημες διαχωριστικές γραμμές.
- Ο μετασχηματισμός περιστροφής για καθεμία από τα winds δίνεται από τον πίνακα:

$$R(\theta)=\begin{vmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{vmatrix}$$

όπου θ είναι η γωνία περιστροφής.

- Περιοδικότητα και Στάθερη Βάση: Ο σταθερός ρυθμός αλλαγής της γωνίας (με βάση το πλήρες περίοδό της περιστροφής) δημιουργεί ομαλή και επαναλαμβανόμενη κίνηση. Η ρύθμιση των offsets διασφαλίζει ότι η βάση του ανεμοπτέρου παραμένει σταθερή στο φόντο.

Με αυτό το πρόγραμμα, επιτυγχάνεται ένα ομαλό βίντεο όπου οι φτερούγες του ανεμοπτέρου περιστρέφονται φυσικά πάνω από το φόντο, χρησιμοποιώντας μαθηματικούς μετασχηματισμούς περιστροφής και μάσκες για ρεαλιστική ενσωμάτωση:
Το output μπορειτε να το δείτε ειτε στο βίντεο με όνομα windmill_rot.avi είτε μέσω της συνάρτηση implay και να το δειτε μεσω του αρχείου παράλληλα. Ενδεικτικό screenshot στο frame 200.

Ερώτημα 6:

Η επανάληψη του προηγούμενου ερωτήματος με διαφορετικές μεθόδους παρεμβολής (nearest, linear/bilinear, bicubic) επικεντρώνεται στη σύγκριση της ποιότητας της εικόνας κατά την περιστροφή των wind.
Τα μαθηματικά πίσω από κάθε μέθοδο υποδεικνύουν πώς υπολογίζονται οι νέες τιμές pixel και πως επηρεάζουν το οπτικό αποτέλεσμα ενός γεωμετρικού μετασχηματισμού

Βασικά Βήματα στον Κώδικα

1. Φόρτωση και Προεπεξεργασία Εικόνων:
    - **Αντίστοιχη φόρτωση εικόνας , μασκας κλπ με πρίν.**

2. Περιστροφή με Διαφορετικές Μεθόδους Παρεμβολής:
    - Για κάθε μέθοδο (nearest, bilinear, bicubic):
        - Η εικόνα περιστρέφεται χρησιμοποιώντας τη συνάρτηση imrotate με την αντίστοιχη επιλογή παρεμβολής.
        - Η μάσκα περιστρέφεται μόνο με nearest για διατήρηση των καθαρών άκρων.

```
 % Define interpolation methods to compare
    interp_methods = {'nearest', 'bilinear', 'bicubic'};
    output_names = {'windmill_rot_nearest.avi', 'windmill_rot_linear.avi',
'windmill_rot_cubic.avi'};

    % Total number of frames and angle step
    n_frames = 200;
    a_step = -360 / n_frames;

    for m = 1:length(interp_methods)
        interp_method = interp_methods{m};
        vid_name = output_names{m};

        vid = VideoWriter(vid_name);
        vid.FrameRate = 30;
        open(vid);

        for k = 1:n_frames
            angle = (k - 1) * a_step;

            % Rotate using chosen interpolation for the rotor
            % Mask remains with nearest to preserve binary nature
            rotor_rot = imrotate(rotor_r, angle, interp_method, 'crop');
            mask_rot = imrotate(mask_r, angle, 'nearest', 'crop');
```

```
            % Overlay on background
            frame = bg_r;
            roi = frame(y_off+1:y_off+new_h, x_off+1:x_off+new_w, :);

            % Blend using mask
            for ch = 1:3
                roi(:,:,ch) = rotor_rot(:,:,ch) .* mask_rot + roi(:,:,ch) .*
(1 - mask_rot);
            end

            frame(y_off+1:y_off+new_h, x_off+1:x_off+new_w, :) = roi;

            writeVideo(vid, im2uint8(frame));


        end
```

3. Δημιουργία Βίντεο:
    - Για κάθε μέθοδο παρεμβολής, τα καρέ αποθηκεύονται σε ξεχωριστό βίντεο: windmill_rot_nearest.avi, windmill_rot_linear.avi, windmill_rot_cubic.avi.

Μαθηματική Προσέγγιση

- Η μέθοδος **Nearest Neighbor** αναθέτει σε κάθε νέο pixel την τιμή του πλησιέστερου γειτονικού pixel της αρχικής εικόνας, ακολουθώντας τον κανόνα $x' = \text{round}(x), y' = \text{round}(y)$. Παράγει απότομα αποτελέσματα, με εμφανείς τετραγωνισμένες ακμές και περιορισμένη ομαλότητα. Είναι η ταχύτερη μέθοδος, αλλά η ποιότητα είναι χαμηλή, καθώς οι αλλοιώσεις γίνονται ιδιαίτερα αισθητές σε κινούμενα στοιχεία, όπως οι φτερωτές ενός ανεμόμυλου.
- Η **Bilinear Interpolation** υπολογίζει τη νέα τιμή κάθε pixel ως σταθμισμένο μέσο όρο των τεσσάρων γειτονικών pixels. Αυτό έχει ως αποτέλεσμα ομαλότερες μεταβάσεις στα χρώματα και λιγότερες εμφανείς αλλοιώσεις. Η μέθοδος bilinear μειώνει την εμφάνιση απότομων αλλαγών και τετραγωνισμένων ακμών, ενώ χρησιμοποιείται ευρέως σε επεξεργασία εικόνας και βίντεο. Σε εφαρμογές με κίνηση, διατηρεί πιο φυσικές μεταβάσεις, κάνοντας τις περιστροφές και τις αλλαγές μεγέθους πιο ομαλές σε σύγκριση με το nearest.
- Η **Bicubic Interpolation** χρησιμοποιεί τις τιμές των 16 γειτονικών pixels και μια σειρά από spline συναρτήσεις για την εκτίμηση των νέων τιμών. Αυτή η μέθοδος προσφέρει την υψηλότερη ποιότητα εικόνας, με εξαιρετικά ομαλές μεταβάσεις και λεπτομερείς δομές, χωρίς εμφανείς αλλοιώσεις ή απώλειες ευκρίνειας. Αν και είναι πιο αργή σε σύγκριση με τις άλλες μεθόδους, το αποτέλεσμα είναι αισθητά καλύτερο, διατηρώντας φυσικές περιστροφές και λεπτομέρειες χωρίς παραμορφώσεις.

Συμπεράσματα από τη Σύγκριση

- Nearest:
  Γρήγορη επεξεργασία αλλά με χαμηλή ποιότητα εικόνας. Εμφανίζονται blocky artifacts και σκληρά άκρα.
- Bilinear:
  Παρέχει ομαλές μεταβάσεις και μειώνει τα artifacts σε σχέση με τη nearest, αλλά οι λεπτομέρειες γίνονται ελαφρώς απαλές.
- Bicubic:
  Παράγει την καλύτερη ποιότητα, με εξαιρετικά ομαλή περιστροφή και λεπτομερείς μεταβάσεις, αν και απαιτεί περισσότερους υπολογισμούς.

Με την προσέγγιση αυτή, παρατηρούμε πώς οι διαφορετικές μέθοδοι παρεμβολής επηρεάζουν το τελικό οπτικό αποτέλεσμα ενός γεωμετρικού μετασχηματισμού. Η εφαρμογή κάθε μεθόδου δίνει έμφαση στην επιλογή μεταξύ ταχύτητας και ποιότητας.

Οι διαφορες με τη παραπάνω υλοποίηση είναι οπτικα μικρές και μπορεί να εχει γίνει καποιο λάθος στη προσέγγιση .
Μπορειτε να δειτε τα αποτελεσματα:

windmill_rot_nearest.avi, windmill_rot_linear.avi, windmill_rot_cubic.avi

Ερώτημα 7

Το πρόγραμμα που παρατίθεται δημιουργεί ένα βίντεο με βάση τις εικόνες ball.jpg, ball_mask.jpg και beach.jpg, όπου μια μπάλα κινείται πάνω σε παραθαλάσσιο τοπίο με περιστροφές, κλιμακώσεις και μετατοπίσεις.

Κύρια Βήματα και Λογική:

1. **Φόρτωση και Προεπεξεργασία:**
   - Φορτώνονται οι εικόνες της μπάλας, της μάσκας της μπάλας και του παραλιακού φόντου. `ball = im2double(imread('ball.jpg'));`

   - Η μάσκα αντιστρέφεται ώστε να επιλέγονται το μέρος της μπάλας που θα συνδυαστούν με το φόντο.

```
% Invert and pad mask
mask = ~mask;
padSize = 50;
ball = padarray(ball, [padSize padSize], 'replicate');
mask = padarray(mask, [padSize padSize], 'replicate');
```

2. **Αναπροσαρμογή:**
   - Με τη χρήση της μάσκας εντοπίζεται το πλαίσιο της μπάλας (bounding box) για απομόνωση της περιοχής ενδιαφέροντος.(region of interest)POI .
   - Η μπάλα και η μάσκα μειώνονται σε 120×120 pixels για αποδοτικότητα.
   - Το φόντο (beach.jpg) αναπροσαρμόζεται ώστε να καλύπτει ολόκληρο το σκηνικό (600×1080 pixels).

```
% Resize images
ball = imresize(ball, [120 120]);
mask = imresize(mask, [120 120]);
beach = imresize(beach, [600 1080]);
[sceneH, sceneW, ~] = size(beach);

% Store frames for implay
frames = cell(1, 240);
```

   - Η εικόνα της μπάλας και η μάσκα μένουν μέσα στο πλαίσιο αυτό, και στη συνέχεια γίνεται αναπροσαρμογή (resize) για αποδοτικότερη επεξεργασία.

```
% Animation parameters
posX = 120;
posY = sceneH - 200;
velY = -12;
velX = 3;
angle = 0;
```

3. **Προσδιορισμός Τοποθεσίας:**
   - Ορίζεται ένα αρχικό σημείο (posX, posY) στο φόντο beach όπου θα τοποθετηθεί η μπάλα.
   - Υπολογίζονται μετατοπίσεις ώστε η μπάλα να τοποθετηθεί ορθά πάνω στο φόντο.
   - Η ενημέρωση της θέσης της μπάλας σε κάθε καρέ γίνεται ως εξής:

$$posX\_new = posX\_old + velX$$
$$posY\_new = posY\_old + velY$$

```
% Update position and rotation
    posX = posX + velX;
    angle = angle + 2.5;  % Constant rotation
```

4. **Φυσική Κίνηση και Περιστροφή:**
   - Σε κάθε καρέ του βίντεο εφαρμόζονται:
     - **Φυσικές Δυνάμεις:** Προσομοιώνονται η βαρύτητα και μια αναπήδηση (bounce) ώστε η κίνηση της μπάλας να μοιάζει φυσική.
     - **Περιστροφή:** Η γωνία περιστροφής της μπάλας ενημερώνεται σταθερά κάθε καρέ, ώστε η μπάλα να περιστρέφεται συνεχώς.

```
 % Update physics
velY = velY + 0.3;  % Gravity
posY = posY + velY;

% Bounce check
if posY > sceneH - 200
posY = sceneH - 200;
velY = -abs(velY) * 0.8;  % Bounce with energy loss
end
```

5. **Blending (Συγχώνευση) στο Φόντο:**

   - Σε έναν βρόχο 240 επαναλήψεων:
     - Ενημερώνονται τα φυσικά μεγέθη: η κάθετη ταχύτητα τροποποιείται με βαρύτητα, τοποθετήσεις και περιστροφή ενημερώνονται.
     - Η μπάλα περιστρέφεται (imrotate) με γωνία angle, και η αντίστοιχη μάσκα περιστρέφεται.
     - Υπολογίζεται το Region of Interest (ROI) για την τοποθέτηση της περιστρεφόμενης μπάλας πάνω στο φόντο.
     - Με τη χρήση της μάσκας, γίνεται το blending της περιστρεφόμενης μπάλας με το φόντο, δημιουργώντας ένα ομαλό τελικό καρέ.
     - Το καρέ γράφεται στο βίντεο και αποθηκεύεται σε ένα cell array για προεπισκόπηση.

```
% Rotate ball and mask
rotBall = imrotate(ball, angle, 'bilinear');
rotMask = imrotate(mask, angle, 'bilinear');
[rotH, rotW, ~] = size(rotBall);

% Calculate ROI (Region of Interest)
topY = max(1, round(posY - rotH/2));
leftX = max(1, round(posX - rotW/2));
botY = min(sceneH, topY + rotH - 1);
rightX = min(sceneW, leftX + rotW - 1);

% Crop and blend
roiH = botY - topY + 1;
roiW = rightX - leftX + 1;
ballCrop = rotBall(1:roiH, 1:roiW, :);
```

```
maskCrop = rotMask(1:roiH, 1:roiW);

% Create frame and apply blending
frame = beach;
roi = frame(topY:botY, leftX:rightX, :);
```

- Η περιστρεφόμενη μπάλα και η μάσκα συνδυάζονται με το φόντο χρησιμοποιώντας τη μάσκα για να εξασφαλιστεί ότι οι μη-επικαλυπτόμενες περιοχές του φόντου παραμένουν αμετάβλητες.
    - Μαθηματικά, για κάθε pixel:

$$\text{Final Pixel} = \text{ballPixel} \times \text{mask} + \text{backgroundPixel} \times (1 - \text{mask})$$

```
% Blend each channel
for ch = 1:3
    roi(:,:,ch) = ballCrop(:,:,ch).*maskCrop + roi(:,:,ch).*(1-maskCrop);
end
```

εξασφαλίζοντας φυσική ενσωμάτωση της περιστρεφόμενης μπάλας στο τοπίο.

6. **Δημιουργία και Αποθήκευση Βίντεο:**

- Όταν η μπάλα χτυπά το κάτω μέρος της σκηνής, ελέγχουμε εάν η τιμή posY ξεπερνά ένα όριο που αντιπροσωπεύει το έδαφος (π.χ., sceneH - 200). Αν συμβεί αυτό, αντιστρέφουμε την ταχύτητα

```
if posY > sceneH - 200
posY = sceneH - 200;
velY = -0.8 * velY;
end
```

- Ο βρόχος τελειώνει, το VideoWriter κλείνει, και το βίντεο αποθηκεύεται με όνομα transf_beach.avi.
- Η αναπαραγωγή του animation εκτελείται με implay.

Μαθηματική Εστίαση:

- **Affine Μετασχηματισμός & Περιστροφή:** Η εφαρμογή του imrotate βασίζεται σε affine μετασχηματισμούς που περιστρέφουν την εικόνα γύρω από το κέντρο της με μαθηματικά ομαλές μεταβάσεις.
- **Κλιμάκωση & Μετατόπιση:** Η κλιμάκωση των εικόνων μέσω imresize και η μετατόπιση της μπάλας στον καμβά βασίζονται σε γραμμικούς μετασχηματισμούς και αλγεβρικούς υπολογισμούς θέσης.
- **Blending:** Η χρήση της μάσκας για το blending εφαρμόζει απλές γραμμικές εξισώσεις για την ανάμειξη δύο εικόνων, δίνοντας φυσικά αποτελέσματα στην ενσωμάτωση του περιστρεφόμενου αντικειμένου πάνω στο φόντο.

Συμπέρασμα: Το πρόγραμμα συνδυάζει μαθηματικούς μετασχηματισμούς περιστροφής, κλιμάκωσης και μετατόπισης με φυσικές προσομοιώσεις κίνησης και χρήση μάσκας για να δημιουργήσει ένα ρεαλιστικό και ομαλό animation μιας περιστρεφόμενης μπάλας πάνω σε παραλία.

Μπορείτε να δείτε το αποτέλεσμα της κίνησης στο transf_beach.avi.

Ερώτημα 8:

Ανάλυση και Εξισώσεις από τον Κώδικα
Για να τροποποιηθεί η πορεία της μπάλας ώστε να κατευθύνεται προς τη θάλασσα και να εκφυλίζεται στον ορίζοντα, ακολούθησαν τα εξής βήματα:

Ορισμός Νέων Παραμέτρων Κίνησης

- Αρχικές και Τελικές Θέσεις:

```
startX = sceneW * 0.25;
startY = sceneH - 100;
endX = sceneW * 0.5;
endY = 100;
```

Οι τιμές startX και startY ορίζουν τις αρχικές συντεταγμένες, ενώ οι endX και endY καθορίζουν το σημείο στον ορίζοντα.

- Κλίμακα (Μέγεθος):

```
startSize = 200;
endSize = 5;
```

Η μπάλα ξεκινά με διάμετρο startSize και μειώνεται σταδιακά σε endSize.

Υπολογισμός Αλλαγών κατά την Κίνηση

- Μεταβολές ανά καρέ:

```
deltaX = (endX - startX) / totalFrames;
deltaY = (endY - startY) / totalFrames;
deltaSize = (endSize - startSize) / totalFrames;
```

Βρόχος Animation για τη Νέα Διαδρομή

1. Ενημέρωση Θέσης και Μεγέθους:
   Σε κάθε καρέ υπολογίζονται οι νέες τιμές για τη θέση και το μέγεθος:

```
posX = startX + deltaX * i;
posY = startY + deltaY * i;
currentSize = max(1, startSize + deltaSize * i);
```

2. Εφαρμογή Κλίμακωσης και Περιστροφής:
   Η μπάλα και η μάσκα κλιμακώνονται και περιστρέφονται:

```
ballResized = imresize(ball, [round(currentSize) round(currentSize)]);
maskResized = imresize(mask, [round(currentSize) round(currentSize)]);
ballRotated = imrotate(ballResized, angle, 'bilinear');
maskRotated = imrotate(maskResized, angle, 'bilinear');
```

   Η γωνία περιστροφής αυξάνεται σταθερά:
   angle = i * 1.5;
3. Υπολογισμός Τοποθεσίας και Blending:
   - Υπολογίζεται το ROI:

```
topY = max(1, round(posY - rotH / 2));
leftX = max(1, round(posX - rotW / 2));
botY = min(sceneH, topY + rotH - 1);
rightX = min(sceneW, leftX + rotW - 1);
```

   Γίνεται blending με χρήση της μάσκας:

```
roi(:,:,ch) = ballCrop(:,:,ch).*maskCrop + roi(:,:,ch).*(1-maskCrop);
```

4. Αποθήκευση Καρέ:
    - Το ενημερωμένο καρέ γράφεται στο αρχείο:

```
writeVideo(v, im2uint8(frame));
```

Τελικές Ρυθμίσεις

- Αποθήκευση Βίντεο: Το βίντεο αποθηκεύεται ως transf_beach_receding.avi:

```
v = VideoWriter('transf_beach_receding.avi');
v.FrameRate = 30;
open(v);
close(v);
```

Αναπαραγωγή:

```
implay('transf_beach_receding.avi');
```

Στόχος

- Η κίνηση της μπάλας ακολουθεί μια προσαρμοσμένη πορεία προς τον ορίζοντα, συνδυάζοντας μετατόπιση, περιστροφή και κλίμακα.
- Οι υπολογισμοί deltaX, deltaY, deltaSize δημιουργούν μια ομαλή και ελεγχόμενη μετάβαση.
- Η διαδικασία περιστροφής, κλιμακωσης και blending καταδεικνύει τη χρήση γεωμετρικών μετασχηματισμών για τη δημιουργία ρεαλιστικών animations.

Αντίστοιχο επιτυχές βίντεο αποθηκεύεται όπως αναφέρθηκε στο transf_beach_receding.avi'.
Ενδεικτικά Screenshot από το avi:

First frames

Last frames