Άσκηση 2
Θέματα Όρασης Υπολογιστών

Περιεχόμενα

Διαδικασία 1 2
Διαδικασία 2 3
Διαδικασία 3 4
Διαδικασία 4 5
Διαδικασία 5 7
Διαδικασία 6 8
Διαδικασία 7 9
Διαδικασία 8 10

Ο κώδικας μπορεί να βρεθεί εδώ:

https://github.com/GrigorisTzortzakis/Computer-Vision/tree/main/Exercise%202

Διαδικασία 1

Η συνάρτηση imread χρησιμοποιείται για την ανάγνωση εικόνων από αρχεία σε μια μεταβλητή περιβάλλοντος του MATLAB. Μπορεί να διαχειριστεί διάφορες μορφές αρχείων, όπως JPEG, PNG και TIFF. Η έξοδος της συνάρτησης είναι συνήθως ένας πίνακας δισδιάστατος (για ασπρόμαυρες εικόνες) ή τρισδιάστατος (για έγχρωμες εικόνες) που αντιστοιχεί στο περιεχόμενο των pixel της εικόνας.

Η συνάρτηση imwarp εφαρμόζει γεωμετρικούς μετασχηματισμούς σε εικόνες, χρησιμοποιώντας ένα αντικείμενο μετασχηματισμού, όπως τα affine2d ή projective2d. Μέσω αυτού, μπορούμε να πραγματοποιήσουμε κλιμάκωση, περιστροφή, μετάθεση ή σύνθετους προσανατολισμούς της εικόνας, ακολουθώντας τις παραμέτρους που ορίζονται στον αντίστοιχο μετασχηματισμό. Επιπλέον, επιτρέπει τον καθορισμό του τρόπου παρεμβολής (interpolation) και τη διατήρηση συγκεκριμένων ορίων ή διαστάσεων για την έξοδο.

Η συνάρτηση affine2d αντιστοιχεί σε έναν μετασχηματισμό συγγένειας δύο διαστάσεων. Αυτοί οι μετασχηματισμοί περιλαμβάνουν πράξεις όπως μεταθέσεις, κλίμακες και περιστροφές οι οποίες διατηρούν την ευθυγραμμία των σημείων και τις αναλογίες μεταξύ των ευθειών.

Η συνάρτηση projective2d περιγράφει έναν μετασχηματισμό προβολής σε δύο διαστάσεις. Σε αντίθεση με τον συγγένειας, επιτρέπει τη μετατροπή παράλληλων ευθειών σε μη παράλληλες, προσομοιώνοντας έτσι προοπτικές παραμορφώσεις ή κλίσεις σε εικόνες.

Η συνάρτηση imref2d ορίζει ένα αντικείμενο που περιγράφει το σύστημα συντεταγμένων μιας εικόνας, το οποίο χρησιμοποιείται για να προσδιορίσουμε τα φυσικά ή πραγματικά όρια και μεγέθη μετά από μετασχηματισμούς. Με αυτό μπορούμε να καθορίσουμε, για παράδειγμα, την ανάλυση ή την έκταση της εικόνας στον χώρο (όπως pixels ανά μονάδα μήκους) καθώς και τον τρόπο με τον οποίο το MATLAB θα υπολογίσει νέες συντεταγμένες μετά την εφαρμογή ενός μετασχηματισμού.

Τέλος, η συνάρτηση implay επιτρέπει την αναπαραγωγή ακολουθιών εικόνων ή βίντεο. Εφόσον έχουμε μια σειρά από καρέ (frames) που αντιπροσωπεύουν διαφορετικές στιγμές στον χρόνο, μπορούμε μέσω του implay να τα προβάλλουμε διαδοχικά, δημιουργώντας έτσι ένα κινούμενο αποτέλεσμα.

Διαδικασία 2

Αρχικά, επιλέγουμε την εικόνα που μας δίνεται (την παραλία) και αυτή διαβάζεται στη μεταβλητή originalImage μέσω της συνάρτησης imread, ώστε να μπορούμε να επεξεργαστούμε το περιεχόμενό της. Στη συνέχεια, δημιουργείται ένας καμβάς ίδιου μεγέθους με την αρχική εικόνα, ο οποίος αρχικοποιείται με μηδενικές τιμές (μαύρο φόντο). Ορίζονται πέντε συντελεστές κλιμάκωσης (scalingFactors) σε φθίνουσα σειρά, ξεκινώντας από το 1.0 (μη κλιμακωμένη εικόνα) έως το 0.2 (μικρή εικόνα), καθώς και πέντε θέσεις (στοιχεία του πίνακα positions), στις οποίες θα τοποθετηθούν διαδοχικά οι κλιμακωμένες εικόνες. Αυτές οι θέσεις αποτελούνται από συντεταγμένες x, y που προσδιορίζουν το επάνω-αριστερό σημείο εισαγωγής της κάθε κλιμακωμένης εικόνας πάνω στον καμβά.

Η κλιμάκωση πραγματοποιείται με την συνάρτηση affine2d, όπου ο πίνακας [scale 0 0; 0 scale 0; 0 0 1] ορίζει μια ομοιόμορφη κλιμάκωση (ίδιος λόγος κλιμάκωσης καθ' ύψος και κατά πλάτος). Στη συνέχεια, με τη συνάρτηση imwarp εφαρμόζεται ο μετασχηματισμός στην αρχική εικόνα, ώστε να παραχθεί η αντίστοιχη κλιμακωμένη εκδοχή (scaledImage). Επιπλέον, υπολογίζονται τα ορθά όρια (rowEnd, colEnd) για να αποφευχθεί υπέρβαση των διαστάσεων του καμβά σε περίπτωση που η κλιμακωμένη εικόνα ξεπερνά το όριο. Αφού προσδιοριστούν τα όρια, αντιγράφουμε το αντίστοιχο τμήμα της κλιμακωμένης εικόνας στην επιθυμητή θέση του canvas. Με αυτόν τον τρόπο, οι εικόνες τοποθετούνται έτσι ώστε να καλύπτουν διαφορετικές περιοχές.

Τέλος, εμφανίζεται το αποτέλεσμα στον χρήστη μέσω της imshow. Έτσι, παρατηρούμε ότι το τελικό αποτέλεσμα περιλαμβάνει διαφορετικές εκδοχές της ίδιας αρχικής εικόνας, τοποθετημένες σε διάφορα σημεία και με κλιμάκωση που μειώνεται σε κάθε βήμα, ενώ φροντίζουμε να τοποθετήσουμε τις εικόνες ώστε να μην υπάρχουν κενά πουθενά.

Διαδικασία 3

Αρχικά, διαβάζουμε την εικόνα pudding με την συνάρτηση imread και μετα δημιουργούμε ένα αντικείμενο VideoWriter με όνομα εξόδου pudding_erotima3.avi και ρυθμό καρέ FrameRate=30, το οποίο ανοίγουμε για να ετοιμαστούμε να γράψουμε δεδομένα σε μορφή βίντεο. Για την αλληλουχία των εικόνων, ορίζουμε παράγοντες που ελέγχουν τη στρέβλωση. Συγκεκριμένα, η μεταβλητή maxShear=0.2 καθορίζει το εύρος της μέγιστης στρέβλωσης, ενώ η numFrames=120 υποδεικνύει πόσα καρέ θα παραχθούν.

Στη συνέχεια, υπολογίζουμε το μέγεθος της αρχικής εικόνας (height, width) και δημιουργούμε έναν καμβά) τριπλάσιου πλάτους (canvasWidth = width * 3) και ίδιου ύψους (canvasHeight = height) για να έχει χώρο η στρεβλωμένη εικόνα να μετακινείται οπτικά μέσα στο πλάνο.

Παρακάτω, ορίζουμε κάθετες γραμμές (verticalLines) για να τονίσουμε το όριο της εικόνας μετά την εφαρμογή της στρέβλωσης. Αρχικά, παράγουμε ένα περίγραμμα (boundary) βασισμένο στο κανάλι άλφα, έτσι ώστε να γνωρίζουμε ποια σημεία της εικόνας είναι ενεργά (όχι κενά) και να μπορούμε πάνω σε αυτά τα σημεία να δημιουργήσουμε μικρά μαύρα κάθετα segments. Αυτά τα segments γίνονται με την βοήθεια του βρόχου for που ελέγχει αν ο δείκτης x βρίσκεται σε ζυγό αριθμό, προσδιορίζοντας εκεί κεντραρισμένες μικρές κάθετες γραμμές.

Στον κεντρικό βρόχο for i=1:numFrames, σε κάθε επανάληψη, αρχικοποιούμε ένα λευκό φόντο (frame) στο μέγεθος του καμβά. Υπολογίζουμε τον τρέχοντα παράγοντα στρέβλωσης shearFactor ως maxShear×sin(2πi/numFrames) δημιουργώντας έτσι μια περιοδική κίνηση που ακολουθεί το ημιτονοειδές (sin) και συνεπάγεται ότι η εικόνα θα εκτελεί μία ταλάντωση δεξιά-αριστερά.

Για να εφαρμόσουμε την γεωμετρική στρέβλωση, κατασκευάζουμε έναν πίνακα μετασχηματισμού (affine transformation matrix) μέσω του affine2d([1 0 0; shearFactor 1 0; width*1.5 0 1]). Εδώ, ο όρος shearFactor στον πάνω δεξιό υποπίνακα ορίζει την παράμετρο της στρέβλωσης, ενώ το width*1.5 στο στοιχείο μετάφρασης (translation) στον άξονα x δημιουργεί την εντύπωση ότι η στρεβλωμένη εικόνα ξεκινά πιο δεξιά. Κατόπιν, εφαρμόζουμε το imwarp στην αρχική εικόνα, στο κανάλι άλφα, καθώς και στην εικόνα των γραμμών (verticalLines), ορίζοντας παράλληλα μία OutputView (imref2d) ώστε το αποτέλεσμα να έχει τις σωστές διαστάσεις.

Τέλος, το βήμα της σύνθεσης (compositing) πραγματοποιείται αντιγράφοντας το αποτέλεσμα της στρεβλωμένης εικόνας πάνω στο λευκό φόντο, χρησιμοποιώντας το alphaMask που προέρχεται από το κανάλι άλφα. Έτσι, όπου υπάρχει μηδενικό άλφα δεν ζωγραφίζουμε την εικόνα (παραμένει λευκό), ενώ όπου το άλφα είναι πλήρες, αντιγράφεται το κατάλληλο τμήμα της χρωματικής πληροφορίας. Τότε, πάνω στην ήδη συνδυασμένη εικόνα, προσθέτουμε τις κατακόρυφες γραμμές θέτοντας τα αντίστοιχα pixel σε μαύρο χρώμα.

Η κίνηση που επιλέξαμε εμφανίζεται στις παραπάνω εικόνες. Μετακινείτε το κάτω τμήμα προς τα δεξιά, επιστρέφει στο κέντρο, πάει αριστερά και τερματίζει στο κέντρο, δηλαδή στην θέση εκκίνησης.

Διαδικασία 4

Ορίζουμε ένα αντικείμενο εγγραφής βίντεο (VideoWriter) με ρυθμό καρέ 60 και διάρκεια 180 καρέ (numFrames=180), δίνοντας έτσι 3 δευτερόλεπτα αποθηκευμένης κίνησης σε 60 fps. Η παράμετρος maxShear=0.3 καθορίζει το μέγιστο πλάτος στρέβλωσης. Με τη μεταβλητή totalCycles=3 ορίζουμε πόσες πλήρεις ταλαντώσεις (κύκλους στρέβλωσης) θα πραγματοποιήσει η εικόνα μέσα στο συνολικό χρονικό διάστημα.

Προσθέτοντας, για τη δημιουργία της οπτικής βάσης, ανιχνεύουμε πρώτα το περίγραμμα της εικόνας με βάση το κανάλι άλφα (αν το σημείο είναι διαφανές ή όχι) και πάνω σε αυτό το περίγραμμα δημιουργούμε κάθετες γραμμές σε προκαθορισμένες στήλες. Έτσι τονίζουμε καλύτερα το όριο του αντικειμένου όταν αυτό στρεβλώνεται.

Παρακάτω, πριν από την κύρια επαναληπτική διαδικασία, φτιάχνουμε έναν χάρτης ύψους (Y) που έχει τιμή 0 στο κάτω μέρος της εικόνας και φτάνει έως το 1 στην κορυφή. Αυτό επιτρέπει να ελέγξουμε πόσο ισχυρή είναι η στρέβλωση σε κάθε σειρά (γραμμή pixel) καθ' ύψος, έτσι ώστε στις χαμηλές τιμές (βάση της εικόνας) ο οριζόντιος άξονας να μένει σχεδόν αμετάβλητος, ενώ ψηλότερα η στρέβλωση να γίνεται πιο έντονη.

Επιπροσθέτως, υλοποιούμε τον βρόχο που παράγει τα καρέ (i=1...numFrames). Για κάθε βήμα, υπολογίζεται ο τρέχων παράγοντας στρέβλωσης currentShear ως $maxShear * \sin(2\pi * totalCycles * \frac{i}{numFrames}$.

Αυτό δημιουργεί μια ημιτονοειδή ταλάντωση δεξιά-αριστερά, με totalCycles πλήρεις επαναλήψεις μέσα στην περίοδο των 180 καρέ.

Επιπλέον, υλοποιούμε στρέβλωση row by row, δηλαδή για κάθε γραμμή y από το κάτω προς το επάνω μέρος, υπολογίζουμε μια οριζόντια μετατόπιση (shiftAmount) ανάλογα με το ύψος της γραμμής. Όσο πιο ψηλά βρίσκεται η γραμμή (y μικρότερο από το μέγιστο ύψος), τόσο μεγαλύτερη γίνεται η μετατόπιση. Με τον τρόπο αυτό, η βάση της εικόνας (εκεί όπου y≈height) μετακινείται ελάχιστα, ενώ το επάνω τμήμα (y≈1) υφίσταται εντονότερη οριζόντια στρέβλωση.

Τέλος, αφού μετατοπιστούν όλα τα pixel της τρέχουσας γραμμής, συνδυάζουμε το αποτέλεσμα της στρεβλωμένης εικόνας πάνω σε ένα λευκό φόντο, λαμβάνοντας υπόψη το κανάλι άλφα ώστε οι περιοχές που δεν ανήκουν στην εικόνα να παραμείνουν λευκές και προσθέτουμε τις κατακόρυφες γραμμές του ορίου (boundary lines) οι οποίες είναι μαύρες.

Από τις παρακάτω εικόνες βλέπουμε ότι πέτυχαμε το επιθυμητό αποτέλεσμα, αφού η κίνηση εκτελείτε με τον ίδιο τρόπο. Το μόνο διαφορετικό είναι ο αριθμός επαναλήψεων, αλλά αν θέλουμε απλά τον μειώνουμε ώστε να είναι ακριβώς ο ίδιος με αυτόν του παραδείγματος.

Διαδικασία 5

Αρχικά, από τη μάσκα windmill_mask.png, επιλέγουμε μόνο το γκρι κανάλι (με τη rgb2gray) και στη συνέχεια τη μετατρέπουμε σε δυαδική (λευκές περιοχές εκεί όπου είναι κομμένο το σώμα του ανεμόμυλου). Με το ~mask αντιστρέφουμε τη μάσκα, ώστε οι φτερωτές και το σώμα του ανεμόμυλου να έχουν τιμή 1, ενώ το υπόλοιπο να είναι 0. Έπειτα, με την find προσδιορίζουμε τις ακραίες γραμμές και στήλες (bounding box) που περικλείουν το αντικείμενο, ώστε να κόψουμε την περιοχή ενδιαφέροντος από την αρχική εικόνα και τη μάσκα, κρατώντας μόνο τον ανεμόμυλο και το περίγραμμά του.

Στη συνέχεια, εφαρμόζουμε έναν παράγοντα κλιμάκωσης (scale_factor) τόσο στην εικόνα του ανεμόμυλου και τη μάσκα του, όσο και στο φόντο. Με αυτόν τον τρόπο, όλες οι εικόνες αποκτούν ανάλυση που ταιριάζει καλύτερα για το τελικό βίντεο.

Παρακάτω, αφού κλιμακώσουμε το φόντο, δημιουργούμε VideoWriter με ρυθμό 30 fps. Υπολογίζουμε επίσης τις διαστάσεις του φόντου (bg_h, bg_w) και του ανεμόμυλου (obj_h, obj_w), ώστε να μπορούμε να τοποθετήσουμε τον ανεμόμυλο (με τα φτερά του) σε κάποια κεντρική θέση. Για την τοποθέτηση αυτήν, ορίζουμε μεταβλητές αντιστάθμισης y_offset και x_offset, οι οποίες αποτελούν την απαιτούμενη μετατόπιση ώστε ο ανεμόμυλος να εμφανιστεί περίπου στο κέντρο του φόντου.

Επιπροσθέτως, ορίζουμε τον αριθμό των καρέ (num_frames) σε 200 και βρίσκουμε τον ρυθμό περιστροφής από τη μεταβλητή angle_step, όπου ο ανεμόμυλος περιστρέφεται κατά -360 μοίρες συνολικά μέσα σε 200 καρέ. Μέσα στο βρόχο που παράγει το βίντεο, σε κάθε επανάληψη υπολογίζουμε τη γωνία angle και εφαρμόζουμε τη συνάρτηση imrotate τόσο στη βασική εικόνα του ανεμόμυλου όσο και στη μάσκα, ρυθμίζοντας τον τρόπο παρεμβολής. Για τη μάσκα χρησιμοποιούμε nearest ώστε να διατηρούνται οι τιμές 0 και 1, παραμένοντας έτσι σε δυαδική μορφή, ενώ για την εικόνα του ανεμόμυλου χρησιμοποιούμε bicubic για πιο ομαλό οπτικό αποτέλεσμα.

Τέλος, αφού έχουμε περιστρέψει το αντικείμενο και τη μάσκα, αντιγράφουμε το αποτέλεσμα πάνω στο φόντο. Η διαδικασία blending υλοποιείται για κάθε κανάλι (RGB), όπου χρησιμοποιούμε τη μάσκα για να ορίσουμε ποιες περιοχές πρέπει να καλυφθούν από τον ανεμόμυλο και ποιες να παραμείνουν ορατές από το φόντο. Πρακτικά, τα pixel του ανεμόμυλου εμφανίζονται μόνο στις θέσεις όπου η μάσκα είναι 1.

Διαδικασία 6

Στη συγκεκριμένη υλοποίηση, επαναλαμβάνουμε τη διαδικασία της προηγουμένης διαδικασίας, δηλαδή κάνουμε crop τη μάσκα και τον ανεμόμυλο, τους κάνουμε scale και τους περιστρέφουμε, φτιάχνοντας τελικά ένα βίντεο όπου οι φτερωτές κινούνται κυκλικά πάνω στο φόντο. Η διαφορά είναι ότι εδώ δοκιμάζουμε τρεις διαφορετικές μεθόδους παρεμβολής, τις nearest, bilinear και bicubic. Για κάθε μέθοδο παρεμβολής δημιουργούμε ξεχωριστό βίντεο, ώστε να συγκρίνουμε τα αποτελέσματα.

Κατά την κλιμάκωση και την περιστροφή, η μέθοδος παρεμβολής παίζει καθοριστικό ρόλο στην ποιότητα της τελικής εικόνας. Παρατηρούμε ότι nearest αντιστοιχεί στην πιο απλή προσέγγιση, όπου κάθε νέο pixel παίρνει την τιμή του πιο κοντινού παλιού pixel. Αυτό οδηγεί σε αποτέλεσμα με γωνίες και παραμόρφωση στην ευκρίνεια γύρω από τις άκρες. Η μέθοδος bilinear αναλύει τοπικά τέσσερα pixel και υπολογίζοντας το νέο χρώμα ως μέσο όρο των τιμών τους. Έτσι επιτυγχάνεται σαφώς πιο ομαλή εμφάνιση, χωρίς τις έντονες γωνίες, όμως μπορεί να παρατηρηθεί θάμπωμα και απώλεια ευκρίνειας στις ακμές και στα μικρά λεπτομερή στοιχεία. Η μέθοδος bicubic εκτελεί πιο σύνθετους υπολογισμούς, λαμβάνοντας υπόψη μια μεγαλύτερη περιοχή γύρω από κάθε υπό υπολογισμό νέο pixel, χρησιμοποιώντας κυβικές συναρτήσεις παρεμβολής.

Στις παρακάτω εικόνες βλέπουμε τα αποτελέσματα της cubic, linear και nearest αντίστοιχα.

Διαδικασία 7

Αρχικά, για την μάσκα θέλουμε οι τιμές που αντιστοιχούν στην μπάλα να είναι 1 (λευκό) και οι υπόλοιπες 0 (μαύρο), οπότε αντιστρέφουμε την μάσκα (~ball_mask) για να βεβαιωθούμε ότι η μπάλα θα φαίνεται και το φόντο της μάσκας θα απορρίπτεται. Στη συνέχεια, κλιμακώνουμε την εικόνα της μπάλας σε συγκεκριμένο μέγεθος (ορίζεται από το ball_size) και προσθέτουμε περιθώριο (padarray) γύρω της. Με αυτόν τον τρόπο, αποφεύγουμε το φαινόμενο να κόβονται τα άκρα της μπάλας κατά την περιστροφή.

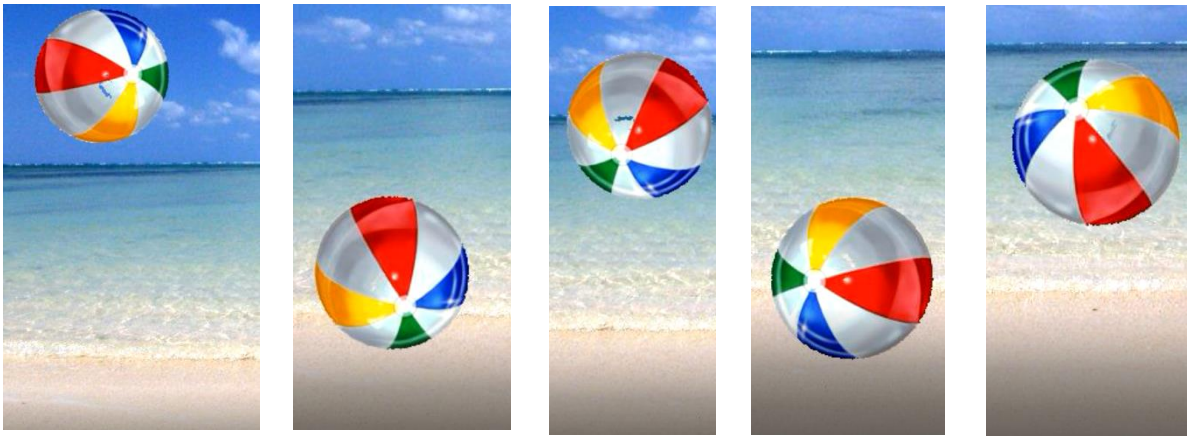
Συνεχίζοντας, για να ορίσουμε το συνολικό μήκος του βίντεο και το πλήθος των καρέ, αξιοποιούμε την παράμετρο fps και τη μεταβλητή duration. Στόχος μας είναι να προκύψουν 30 καρέ ανά δευτερόλεπτο, για δέκα δευτερόλεπτα, άρα συνολικά 300 καρέ. Στη συνέχεια, υπολογίζουμε πού ακριβώς θα εμφανίζεται η μπάλα στο φόντο, επιλέγοντας αρχική και τελική θέση στον οριζόντιο άξονα (start_x και end_x) και ένα ύψος που θεωρούμε έδαφος (ground_y).

Παρακάτω, αναλύουμε την κίνηση της μπάλας σε δύο άξονες. Πρώτον, υπάρχει οριζόντια κίνηση (από τα αριστερά προς τα δεξιά), η οποία ορίζεται γραμμικά από το start_x ως το end_x κατά τη διάρκεια των 300 frame. Δεύτερον, η κατακόρυφη κίνηση επιδιώκει να προσομοιώσει αναπήδηση και για αυτόν τον λόγο, σε κάθε χρονικό διάστημα εφαρμόζουμε μια ημιτονοειδή συνάρτηση sin που δίνει την καμπύλη κίνησης προς τα επάνω και προς τα κάτω. Σε κάθε αναπήδηση, το μέγιστο ύψος μειώνεται με βάση τον συντελεστή απόσβεσης (decay_factor), έτσι ώστε η μπάλα να μοιάζει πως χάνει ενέργεια μετά από κάθε επαφή με το έδαφος.

Επιπλέον, για να είναι η κίνηση πιο ρεαλιστική, κάνουμε την μπάλα να περιστρέφεται περιοδικά γύρω από τον εαυτό της. Η γωνία περιστροφής υπολογίζεται ως angle = -(frame * rotation_speed), όπου το rotation_speed ορίζει πόσες μοίρες περιστρέφεται η μπάλα ανά frame. Η συνάρτηση imrotate στο υλοποιεί αυτήν την περιστροφή, ενώ παράλληλα επιλέγουμε μέθοδο παρεμβολής bilinear για να έχουμε σχετικά ομαλά άκρα κατά την περιστροφή.

Τέλος, αφού περιστρέψουμε την μπάλα (padded_ball), καθώς και την αντίστοιχη μάσκα (padded_mask), υπολογίζουμε τις τελικές συντεταγμένες τοποθέτησης πάνω στην εικόνα του φόντου (τις οποίες περιορίζουμε στην περιοχή που καλύπτει το φόντο, ώστε η μπάλα να μην βγει εκτός οθόνης). Κατόπιν, επιλέγουμε την περιοχή ενδιαφέροντος (ROI) και βάζουμε την περιστρεφόμενη μπάλα σε αυτήν, διατηρώντας μόνο τα pixel που αντιστοιχούν στη μπάλα, χάρη στη μάσκα. Όπου η μάσκα είναι 1, παίρνουμε την πληροφορία χρώματος της μπάλας και όπου είναι 0, αφήνουμε να φανεί το υπόβαθρο της παραλίας.

Παρακάτω μπορούμε να δούμε την κίνηση της μπάλας με το πέρασμα του χρόνου:



Διαδικασία 8

Αρχικά, αντιστρέφουμε την μάσκα της μπάλας ώστε να εξασφαλίσουμε ότι τα σημεία όπου βρίσκεται η μπάλα έχουν λογικές τιμές 1 και το υπόλοιπο καρέ τιμές 0. Στη συνέχεια, προσαρμόζουμε το μέγεθος της εικόνας παραλίας, ώστε να ταιριάζει στις διαστάσεις και έτσι καθορίζουμε ένα canvas συγκεκριμένων διαστάσεων και φροντίζουμε οι μετασχηματισμοί της μπάλας να έχουν λογική κλίμακα πάνω σε αυτόν.

Ακολούθως, ορίζουμε βασικές παραμέτρους όπως fps, τη συνολική διάρκεια (duration) και τον συνολικό αριθμό καρέ (total_frames = fps * duration). Επίσης, προσδιορίζουμε τις αρχικές και τελικές συντεταγμένες της μπάλας στον οριζόντιο και κατακόρυφο άξονα, έτσι ώστε να ξεκινά από ένα σημείο στο προσκήνιο και να καταλήγει πιο μακριά στον ορίζοντα. Για να δώσουμε την εντύπωση ότι η μπάλα μικραίνει όσο απομακρύνεται, ορίζουμε αρχικό μέγεθος σε pixel (initial_ball_size) και ελάχιστο μέγεθος (min_ball_size), και στη συνέχεια υπολογίζουμε μια γραμμική μείωση του μεγέθους σε κάθε frame. Παράλληλα, ορίζουμε την ταχύτητα περιστροφής (rotation_speed), που καθορίζει πόσες μοίρες θα στριφογυρίζει η μπάλα ανά frame, κάνοντας το animation πιο ρεαλιστικό.

Στη συνέχεια, ξεκινάμε τον κεντρικό βρόχο από το πρώτο μέχρι το τελευταίο καρέ (1 έως total_frames). Σε κάθε επανάληψη, αντιγράφουμε αρχικά το φόντο της παραλίας σε μια μεταβλητή current_frame. Έπειτα, υπολογίζουμε το τρέχον μέγεθος της μπάλας με βάση το πόσο έχουμε διανύσει από το αρχικό μέχρι το τελικό στάδιο της κίνησης. Η κλίμακα μικραίνει καθώς περνούν τα καρέ, ώστε να δίνεται η αίσθηση απομάκρυνσης.

Αφού κλιμακώσουμε την εικόνα της μπάλας και τη μάσκα της στο τρέχον μέγεθος, χρησιμοποιούμε τη συνάρτηση padarray για να προσθέσουμε περιθώριο γύρω από τη μπάλα. Αυτό μας εξασφαλίζει ότι η περιστροφή της δεν θα κόψει τα άκρα της εικόνας. Στη συνέχεια, υπολογίζουμε την τρέχουσα γωνία περιστροφής, χρησιμοποιώντας μια συνάρτηση του καρέ και της παραμέτρου rotation_speed. Εφαρμόζουμε τη imrotate για να περιστρέψουμε

τόσο τη μπάλα όσο και τη μάσκα, χρησιμοποιώντας μέθοδο παρεμβολής bilinear. Μετά, καθορίζουμε πού ακριβώς θα τοποθετηθεί η μπάλα στο καρέ, κεντράροντας την περιστρεφόμενη εικόνα στις επιθυμητές συντεταγμένες x και y. Κάνουμε επίσης έλεγχο για να μη βγούμε εκτός ορίων της σκηνής εάν η περιστροφή και η μετάφραση πάει την μπάλα πολύ κοντά στα άκρα.

Τέλος, ορίζουμε ROI μέσα στο current_frame, εκεί όπου θα τοποθετηθεί η περιστρεφόμενη μπάλα. Με μια σύντομη διαδικασία blending για κάθε κανάλι χρώματος (R, G, B) επιλέγουμε τα pixel της μπάλας όπου η μάσκα είναι 1, αντικαθιστώντας έτσι το φόντο με την μπάλα, και διατηρούμε το φόντο όπου η μάσκα είναι 0. Αφού ανανεώσουμε την ROI, την τοποθετούμε πίσω στο current_frame και με την εντολή writeVideo προσθέτουμε το καρέ που προκύπτει στο βίντεο. Με αυτό τον τρόπο, δημιουργούμε σταδιακά όλα τα καρέ όπου η μπάλα μικραίνει και μετατοπίζεται στον άξονα της σκηνής, πηγαίνοντας προς τη θάλασσα.