Γραφική με Υπολογιστές 2023

Εργασία #2: Μετασχηματισμοί και Προβολές

Ζητούμενα

Α. Κλάση Μετασχηματισμών

Να υλοποιηθεί η κλάση:

```
class Transform:
    # Interface for performing affine transformations.
    def __init__(self):
        # Initialize a Transform object.
        self.mat = identity

    def rotate(self, theta: float, u: np.ndarray) -> None:
        # rotate the transformation matrix

    def translate(self, t: np.ndarray) -> None:
        # translate the transformation matrix.

    def transform_pts(self, pts: np.ndarray) -> np.ndarray:
        # transform the specified points
        # according to our current matrix.
```

αντικείμενα της οποίας αντιπροσωπεύουν affine μετασχηματισμούς. Θα έχει ως attribute τον 4×4 πίνακα `mat`, ο οποίος αντιπροσωπεύει έναν affine μετασχηματισμό και θα αρχικοποιείται με $\mathbf{I}_4$.

Ακόμα θα υλοποιεί το παρακάτω interface:

- `rotate(self, theta: float, u: np.ndarray)`: Μέθοδος η οποία υπολογίζει τον πίνακα περιστροφής, που αντιστοιχεί σε δεξιόστροφη περιστροφή κατά γωνία `theta` σε rads περί άξονα με κατεύθυνση που δίνεται από το μοναδιαίο διάνυσμα `u`. Το attribute `mat` θα πρέπει να ενημερώνεται κατάλληλα.

- `translate(self, t: np.ndarray)`: Μέθοδος που ενημερώνει κατάλληλα το attribute `mat` με το διάνυσμα μετατόπισης `t`.

- `transform_pts(self, pts:np.ndarray) -> np.ndarray`: Μέθοδος που μετασχηματίζει τα 3D σημεία `pts` $\in \mathbb{R}^{N \times 3}$ σύμφωνα με τον πίνακα μετασχηματισμού `mat`.

Γ. Συνάρτηση αλλαγής συστήματος συντεταγμένων

Να υλοποιηθεί η συνάρτηση:

```
def world2view(pts: np.ndarray, R: np.ndarray, c0: np.ndarray) -> np.ndarray:
    # Implements a world-to-view transform, i.e. transforms the specified
    # points to the coordinate frame of a camera. The camera coordinate frame
    # is specified rotation (w.r.t. the world frame) and its point of reference
    # (w.r.t. to the world frame).
```

η οποία μετασχηματίζει τα σημεία εισόδου `pts` στο σύστημα συντ/νων της κάμερας. Πιο συγκεκριμένα:

- `pts` $\in \mathbb{R}^{3\times N}$ είναι ο $3 \times N$ πίνακας με τις συντ/νες των αρχικών σημείων ως προς ένα σύστημα συντ/νων.
- `R` $\in \mathbb{R}^{3\times 3}$ είναι ο πίνακας περιστοφής το νέου συστήματος ως προς το αρχικό.
- `c0` $\in \mathbb{R}^3$ το σημείο αναφοράς του νέου συστήματος συντ/νων ως προς το αρχικό.

Η συνάρτηση θα επιστρέφει έναν πίνακα $N \times 3$ με τα σημεία `pts` μετασχηματισμένα ως προς το σύστημα συντ/νων της κάμερας.

Δ. Συνάρτηση προσανατολισμού κάμερας

Να υλοποιηθεί η συνάρτηση:

```
def lookat(eye: np.ndarray, up: np.ndarray, target: np.ndarray) -> Tuple[np.ndarray,
np.ndarray]:
    # Calculate the camera's view matrix (i.e., its coordinate frame transformation
specified
    # by a rotation matrix R, and a translation vector t).
    # :return a tuple containing the rotation matrix R (3 x 3) and a translation
vector
    #   t (1 x 3)
```

Η συνάρτηση θα επιστρέφει τις παραμέτρους που χρειάζονται για το μετασχηματισμό σημείων από το WCS στο σύστημα συντ/νων της κάμερας. Αυτές θα είναι ο πίνακας περιστροφής $R \in \mathbb{R}^{3\times 3}$, και το διάνυσμα μετατόπισης $t \in \mathbb{R}^3$. Πιο συγκεκριμένα:

- `eye` $\in \mathbb{R}^3$ είναι το σημείο του κέντρου της κάμερας.
- `up` $\in \mathbb{R}^3$ είναι το μοναδιαίο up διάνυσμα της κάμερας.
- `target` $\in \mathbb{R}^3$ είναι το σημείο στόχος.

Ε. Συνάρτηση προοπτικής προβολής με pinhole κάμερα

Να υλοποιηθεί η συνάρτηση:

```
def perspective_project(pts: np.ndarray, focal: float, R: np.ndarray, t: np.ndarray)
-> Tuple[np.ndarray, np.ndarray]:
    # Project the specified 3d points pts on the image plane, according to a pinhole
perspective projection model.
```

η οποία παράγει τις προοπτικές προβολές και το βάθος των σημείων εισόδου `pts`. Πιο συγκεκριμένα:

- `pts` $\in \mathbb{R}^{3\times N}$ είναι ο πίνακας των σημείων εισόδου.

- `focal` είναι η απόσταση του πετάσματος της κάμερας από το κέντρο (μετρημένη στις μονάδες που χρησιμοποιεί το σύστημα συντ/νων της κάμερας).
- `R` $\in \mathbb{R}^{3\times3}$ είναι ο πίνακας περιστροφής προς το σύστημα συντ/νων της κάμερας.
- `t` $\in \mathbb{R}^3$ είναι το διάνυσμα μετατόπισης προς το σύστημα συντ/νων της κάμερας.

Η συνάρτηση θα επιστρέφει τις 2D συντ/νες των σημείων εισόδου στο πέτασμα της κάμερας.

ΣΤ. Συνάρτηση απεικόνισης

Να υλοποιήσετε τη συνάρτηση:

```
13  def rasterize(pts_2d: np.ndarray, plane_w: int, plane_h: int, res_w: int, res_h: int)
     -> np.ndarray:
14      # Rasterize the incoming 2d points from the camera plane to image pixel
    coordinates
```

η οποία απεικονίζει τις συντ/νες των σημείων από το σύστημα συντ/νων του πετάσματος της κάμερας, με πέτασμα `plane_h` $\times$ `plane_w`, σε ακέραιες θέσεις (pixels) της εικόνας διάστασης `res_h` $\times$ `res_w`.

Σημείωση: Ο άξονας της κάμερας περνά από το κέντρο ορθογώνιου διάστασης `plane_h` $\times$ `plane_w`, ενώ η αρίθμιση του `res_h` $\times$ `res_w` πίνακα της εικόνας ξεκινά από τα κάτω προς τα πάνω και από αριστερά προς δεξιά και έχει τιμές $[0, \ldots,$ `res_w`$]$ οριζοντίως και $[0, \ldots,$ `res_h`$]$ κατακορύφως.

Ζ. Συνάρτηση φωτογράφισης

Να υλοποιήσετε τη συνάρτηση:

```
15  def render_object(v_pos, v_clr, t_pos_idx, plane_h, plane_w, res_h, res_w, focal, eye
    , up, target) -> np.ndarray:
16      # render the specified object from the specified camera.
```

η οποία φωτογραφίζει μία 3D σκηνή ενός αντικειμένου από μία κάμερα. Πιο συγκεκριμένα:

- `v_pos` $\in \mathbb{R}^{N\times3}$ είναι οι τρισδιάστατες συντ/νες των σημείων του αντικειμένου.
- `v_clr` $\in \mathbb{R}^{N\times3}$ είναι ο πίνακας με τα χρώματα των κορυφών.
- `t_pos_idx` είναι ο πίνακας που περιέχει δείκτες σε σημεία του πίνακα `v_pos` που αποτελούν τις κορυφές των τριγώνων. Ο πίνακας είναι διάστασης $F \times 3$. Η i-οστή γραμμή του πίνακα, δηλώνει τις τρεις κορυφές που σχηματίζουν το τρίγωνο (με αναφορά σε κορυφές του πίνακα `v_pos` και η αρίθμησή του ξεκινάει από το 0).
- `plane_h` και `plane_w` είναι το ύψος και το πλάτος του πετάσματος της κάμερας.
- `res_h` και `res_w` είναι το ύψος και το πλάτος του καμβά σε pixel.
- `focal` είναι η απόσταση του πετάσματος από το κέντρο της κάμερας.
- `eye` είναι το κέντρο της κάμερας ως προς το WCS.
- `up` είναι το up διάνυσμα της κάμερας.
- `target` είναι το σημείο στόχος της κάμερας.

Ολά τα σημεία, και οι πίνακες που αφορούν 3D συντ/νες δίνονται σε μη ομογενή μορφή. Η συνάρτηση θα επιστρέφει έναν πίνακα $res_h \times res_w \times 3$ (τη φωτογραφία του αντικειμένου) και χρησιμοποιώντας κατάλληλα τις παραπάνω συναρτήσεις θα υλοποιεί όλο το pipeline της απεικόνισης ενός αντικειμένου. Επίσης θα πρέπει να χρησιμοποιεί τη συνάρτηση πλήρωσης της προηγούμενης εργασίας για να χρωματίσει το αντικείμενο με τη μέθοδο Gouraud.

Παραδοτέα

1. Οι παραπάνω συναρτήσεις σε μορφή **σχολιασμένου** πηγαίου κώδικα python (μέχρι $v3.10$) με σχόλια γραμμένα στα **αγγλικά** ή **greeklish**, (κοινώς, **μη γράφετε σχόλια με ελληνικούς χαρακτήρες)**.

2. script επίδειξης με όνομα demo.py. Το script θα πρέπει να καλείται χωρίς εξωτερικά ορίσματα, να διαβάζει το αντικείμενο από το αρχείο hw2.npy που σας δίνεται, και να εκτελεί ένα προκαθορισμένο σύνολο μετασχηματισμών, ο οποίος περιγράφεται παρακάτω:

   Ως είσοδο χρησιμοποιείστε τον πίνακα `v_pos`, οποίος περιέχει τις τρισδιάστατες συντεταγμένες των N κορυφών των τριγώνων που αποτελούν το αντικείμενο. Δοθέντων των σημείων του πίνακα `v_pos`, το script σας θα πρέπει να εκτελεί σειριακά τα ακόλουθα βήματα:

   (α΄) Τα περιστρέφει κατά γωνία κάτα `theta` rad περί άξονα παράλληλο προς διάνυσμα κάτα `rot_axis`.

   (β΄) Τα μετατοπίζει κάτα `t_1`.

   (γ΄) Τα μετατοπίζει κατά `t_2`

   Κάθε βήμα θα δέχεται ως είσοδο την έξοδο του προηγούμενου. Μετά από κάθε βήμα θα πρέπει να φωτογραφίζετε το αντικείμενο, καλώντας τη συνάρτηση `v_pos` με παραμέτρους κάμερας `eye, target, up, focal` και να το χρωματίζετε καλώντας τη συνάρτηση `render` της πρώτης εργασίας με τη χρήση Gouraud shading.

   Συνολικά, θα πρέπει να παράξετε 4 φωτογραφίες του αντικειμένου, μία στην αρχική του θέση, και μία για τα αποτελέσματα των βημάτων (α') - (γ'). Κάθε φωτογραφία να αποθηκεύεται με με όνομα αρχείου τον αριθμό του βήματος (θεωρώντας ότι η αρχική θέση είναι το βήμα 0) και επέκταση jpg.

   Σημείωση 1: Αν δεν είχατε υλοποιήσει τη συνάρτηση `render`, ή αν είχατε κάποιο λάθος στην υλοποίησή της, μπορείτε να χρησιμοποιήσετε τη `render` κάποιου/κάποιας συναδέλφου σας, **αρκεί να το δηλώσετε το όνομά του στην αναφορά σας**.

   Σημείωση 2: Το αρχείο hw2.npy που σας δίνεται, περιέχει τις παραμέτρους του αντικειμένου `v_pos, v_clr, t_pos_idx` καθώς και όλες τις παραμέτρους που θεωρούνται γνωστές (παραμέτρους κάμερας, διανύσματα μετατόπισης, άξονες περιστροφής κτλ.).

3. Αναφορά με:

   - Περιγραφή της λειτουργίας και του τρόπου κλήσης των προγραμμάτων.
   - Περιγραφή των συναρτήσεων.
   - Τα ενδεικτικά αποτελέσματα που παράγονται από το demo.

Παρατηρήσεις

- Θεωρείστε ότι: `res_h = res_w = 512`.
- `plane_h = plane_w = 15`
- `focal = 70`
- Μην κάνετε τεχνητές περιστροφές της φωτογραφίας προκειμένου να φαίνεται ”ίσιο” το αντικείμενο.
- Οι εργασίες αξιολογούνται με χρήση Python(ως v3.10).
- Οι εργασίες είναι **αυστηρά** ατομικές.
- Το background του καμβά είναι λευκό `rgb = np.array([1.0, 1.0, 1.0])`.
- Υποβάλετε ένα και μόνο αρχείο, τύπου .zip.
- Το όνομα του αρχείου πρέπει να είναι AEM.zip, όπου AEM είναι τα τέσσερα ψηφία του Α.Ε.Μ. του φοιτητή της ομάδας.
- Το προς υποβολή αρχείο πρέπει να περιέχει τα αρχεία κώδικα python και το αρχείο report.pdf το οποίο θα είναι η αναφορά της εργασίας.
- Η αναφορά πρέπει να είναι ένα αρχείο τύπου PDF, και να έχει όνομα report.pdf.
- Όλα τα αρχεία κώδικα πρέπει να είναι αρχεία κειμένου τύπου UTF-8, και να έχουν κατάληξη .py .
- Το αρχείο τύπου zip που θα υποβάλετε δεν πρέπει να περιέχει κανένα φάκελο.
- Για την ονομασία των αρχείων που περιέχονται στο προς υποβολή αρχείο, χρησιμοποιείτε μόνο αγγλικούς χαρακτήρες, και όχι ελληνικούς ή άλλα σύμβολα, πχ “#”, “$”, “%” κλπ.

Προσοχή: Θα αξιολογηθούν μόνο όσες εργασίες έχουν demos που τρέχουν!