УДК 004.93

Т.І. Присяжнюк

ОСОБЛИВОСТІ ВІЗУАЛІЗАЦІЇ 4-ВИМІРНИХ ОБ'ЄКТІВ ЗАСОБАМИ МОВИ PYTHON

Вінницький національний технічний університет

Анотація:

У тезах розглянуто теоретичні основи та практичні методи комп'ютерної візуалізації чотиривимірних геометричних об'єктів засобами мови програмування Python. Проаналізовано математичний апарат перспективних і ортографічних проєкцій з чотиривимірного простору, наведено ключові формули матриць перетворень. Описано провідні бібліотеки та алгоритми, що застосовуються для побудови 4D-зображень, а також розглянуто конкретні сфери застосування чотиривимірної геометрії – від фізики та машинного навчання до медицини та ігрової індустрії. Окреслено виклики та перспективи розвитку програмних засобів для роботи з багатовимірними просторами.

Ключові слова:

чотиривимірний простір, гіперкуб, тесеракт, проєкція, Python, matplotlib, NumPy, 4D-візуалізація, анімація, зріз, матриця повороту, зниження розмірності.

Abstract:

The paper examines the theoretical foundations and practical methods of computer visualization of four-dimensional geometric objects using the Python programming language. The mathematical framework of perspective and orthographic projections from four-dimensional space is analyzed, and the key transformation matrix formulas are presented. The leading libraries and algorithms used for constructing 4D images are described, and specific application areas of four-dimensional geometry are discussed — from physics and machine learning to medicine and the gaming industry. The challenges and prospects for the development of software tools for working with multidimensional spaces are outlined.

Keywords:

four-dimensional space, hypercube, tesseract, projection, Python, matplotlib, NumPy, 4D visualization, animation, cross-section, rotation matrix, dimensionality reduction.

Вступ

Чотиривимірний простір є одним з центральних понять сучасної математики і теоретичної фізики, однак його наочна інтерпретація залишається складною задачею для людського сприйняття. Ще наприкінці XIX століття Едвін Ебботт у своїй алегоричній повісті [1] наочно продемонстрував, наскільки важко мешканцям простору нижчої вимірності уявити простір вищої. Ця ідея залишається актуальною і в епоху комп'ютерної графіки: попри значний прогрес обчислювальних технологій, побудова інтуїтивно зрозумілого представлення 4D-об'єктів досі є нетривіальним завданням.

Комп'ютерна візуалізація відіграє ключову роль у навчанні та науковому дослідженні: вона дозволяє створювати інтерактивні моделі об'єктів, фізичне існування яких у тривимірному світі є неможливим. Мова програмування Python завдяки своїй гнучкості, лаконічному синтаксису та розвиненій екосистемі наукових бібліотек стала одним із провідних інструментів для реалізації подібних задач. Можливість поєднати символьні обчислення, чисельний аналіз і 3D-рендеринг у межах одного середовища робить Python природним вибором для дослідників, які працюють з багатовимірною геометрією [2].

Сфери застосування чотиривимірної геометрії

Незважаючи на абстрактну природу, 4-вимірний простір знаходить практичне застосування у численних галузях науки та техніки. Розуміння особливостей 4D-геометрії та вміння її візуалізувати є корисним інструментом не лише для математиків, а й для інженерів, фізиків, програмістів і фахівців з

обробки даних.

У теоретичній і прикладній фізиці чотиривимірний простір-час є фундаментальним поняттям спеціальної та загальної теорії відносності. Тут три просторові координати доповнюються часовою координатою ct, утворюючи простір Мінковського. Візуалізація просторово-часових діаграм, конусів світла та геодезичних ліній у загнутому 4D-просторі-часі є важливим інструментом для розуміння релятивістських ефектів. Caplan [3] описує реалізацію інтерактивної візуалізації тетраедральних 4D просторово-часових сіток засобами Python для задач чисельного розв'язку рівнянь Ейнштейна.

У машинному навчанні та аналізі даних практично будь-який реальний набір даних є багатовимірним – ознаковий простір задачі може налічувати десятки, сотні й тисячі вимірів. Методи зниження розмірності (PCA, t-SNE, UMAP) фактично вирішують задачу, зворотну до 4D-візуалізації: вони проєктують дані з простору n-ої вимірності у 2D або 3D-простір, придатний для сприйняття. Порівняльний аналіз цих методів [5] показує, що вибір алгоритму суттєво впливає на збереження як локальної, так і глобальної структури даних, що безпосередньо пов'язано з математикою проєкцій.

В медицині 4D-дані виникають природним чином при роботі з динамічними об'ємними зображеннями (4D = 3D + час): КТ серця в різні фази серцевого циклу, 4D МРТ мозку при функціональних дослідженнях, ультразвукові дослідження плоду в реальному часі. Алгоритми обробки та візуалізації таких даних, реалізовані у Python з використанням бібліотек SimpleITK і VTK, дозволяють медикам досліджувати динамічні процеси у тривимірному анатомічному контексті.

У комп'ютерній графіці та ігровій індустрії 4D-геометрія застосовується у процедурній генерації шуму (4D simplex noise) для створення реалістичних динамічних текстур та ландшафтів, що природно змінюються з часом. Четвертий вимір тут відіграє роль часового або анімаційного параметра, що дозволяє генерувати плавні безшовні переходи. Крім того, розробники деяких ігор (наприклад, Miegakure) повністю побудовані на механіці переміщення у 4D-просторі, що вимагає ефективних алгоритмів проєкції в реальному часі. Математичний апарат проєкцій та поворотів.

Основу 4D-візуалізації становить математика лінійних перетворень. Будь-який 4-вимірний об'єкт описується вектором координат $v = (x, y, z, w)^T$. Для відображення на площину застосовується послідовність двох проєкцій: спочатку з R^4 у R^3, потім з R^3 у R^2.

Перспективна проєкція з R^4 у R^3 виконується за формулою, що масштабує три координати залежно від відстані до «4D-камери» d:

$$x' = \frac{x}{d - w}, \quad y' = \frac{y}{d - w}, \quad z' = \frac{z}{d - w} \quad (1)$$

де d – відстань від «ока» у четвертому вимірі до об'єкту. При $d \to \infty$ формула (1) вироджується в ортографічну проєкцію, де четверта координата просто відкидається. Аналогічно, подальша проєкція отриманого 3D-вектора (x', y', z') на площину зображення виконується за стандартною формулою перспективи:

$$X = \frac{f \cdot x'}{z'}, \quad Y = \frac{f \cdot y'}{z'} \quad (2)$$

де f – фокусна відстань камери. Обидві формули (1) і (2) реалізуються у Python як операції ділення масиву NumPy на скаляр, що забезпечує векторизовану обробку всіх вершин об'єкту одночасно [2].

Обертання у 4D-просторі здійснюється у шести незалежних площинах: XY, XZ, XW, YZ, YW та ZW. Матриця повороту у площині XW на кут θ має вигляд:

$$R_{XW}(\theta) = \begin{bmatrix} \cos\theta & 0 & 0 & -\sin\theta \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sin\theta & 0 & 0 & \cos\theta \end{bmatrix} \quad (3)$$

Аналогічно будуються матриці для решти п'яти площин. Довільний поворот у R^4 є суперпозицією обертань у кількох площинах одночасно – на відміну від R^3, де будь-який поворот зводиться до обертання навколо єдиної осі (теорема Ейлера). Для анімації тесеракту або іншого 4D-об'єкту достатньо покроково збільшувати кут θ в одній або кількох матрицях і перемножувати їх на вектори

вершин:

$$v_i' = R_{XW}(\theta) \cdot R_{YW}(\varphi) \cdot v_i \quad (4)$$

де v_i – вектор координат і-ї вершини об'єкту. Операція множення матриць виконується через np.dot() або оператор @ у Python. Для тесеракту з 16 вершинами всі перетворення застосовуються векторизовано до масиву розміром (16 × 4), що забезпечує обчислення одного кадру анімації за мікросекунди [6].

Інструменти та бібліотеки Python для 4D-візуалізації

Реалізація 4D-візуалізації у Python спирається на кілька ключових бібліотек. Бібліотека NumPy [7] є фундаментальним інструментом: вона забезпечує ефективне виконання матричних операцій для перетворення координат, побудови матриць повороту (3)–(4) та перспективних проєкцій (1)–(2). Зберігання координат усіх вершин об'єкту у вигляді двовимірного масиву NumPy (N × 4) дозволяє застосовувати всі перетворення до всіх вершин одночасно без явних циклів.

Бібліотека Matplotlib [4] дозволяє будувати як статичні 3D-проєкції засобами mpl_toolkits.mplot3d, так і плавні анімації через клас FuncAnimation. Для кожного кадру анімації перераховуються нові координати вершин після збільшення кута θ, після чого відповідні ребра перемальовуються у тривимірних осях. Це дозволяє реалізувати інтерактивне обертання тесеракту у кількох десятках рядків коду.

Для більш продуктивної 3D-графіки та поверхневого рендерингу використовується бібліотека PyVista – уніфікований інтерфейс до бібліотеки VTK. Вона надає зручні інструменти для побудови полігональних сіток 4D-об'єктів після проєкції, а також підтримує апаратне прискорення через OpenGL. Метод зрізів (cross-section), де при фіксованому значенні w знаходиться перетин 4D-сітки з гіперплощиною, описаний у роботі Cavallo [6]: кожен тетраедр, що перетинається гіперплощиною, породжує трикутник або чотирикутний полігон, з яких складається фінальна 3D-поверхня.

Бібліотека Plotly дозволяє будувати інтерактивні 3D-графіки у браузері, що особливо зручно для освітніх веб-застосунків з 4D-геометрії. Колірне кодування четвертої координати реалізується через параметр color у Plotly: кожній точці проєкції призначається колір відповідно до значення w, що дозволяє зберегти інформацію про четвертий вимір у вигляді додаткового візуального каналу. Для задач машинного навчання бібліотека scikit-learn надає реалізації PCA, t-SNE та UMAP – алгоритмів, що фактично вирішують обернену задачу 4D-візуалізації [5].

Виклики та перспективи розвитку

Основним викликом при роботі з 4D-візуалізацією є принципова обмеженість людського сприйняття: жодна проєкція не може передати всю інформацію про 4-вимірний об'єкт без втрат, так само як 2D-малюнок куба не є еквівалентом самого куба. Дослідження Al-Jarrah et al. [8] показало, що різні парадигми візуалізації – проєкції, зрізи, колірне кодування та анімація — суттєво різняться за ефективністю залежно від конкретного завдання та рівня підготовки користувача. Найкращі результати досягаються при комбінуванні кількох методів одночасно.

Ще однією проблемою є вибір зручної системи керування поворотами у 4D-просторі для інтерактивних застосунків. У 3D-просторі для цього зазвичай використовуються кватерніони, які дозволяють уникнути «замикання кардана» та забезпечують плавну інтерполяцію. Аналогом кватерніонів у 4D є ротори алгебри Кліффорда, реалізація яких у Python значно складніша. Ця математика детально розглянута у монографії Hanson [2], де також наведено практичні алгоритми для реалізації у комп'ютерній графіці.

Перспективним напрямом є інтеграція 4D-візуалізації з технологіями доповненої та віртуальної реальності. Використання Python разом із OpenXR або WebXR-фреймворками відкриває можливості для побудови іммерсивних середовищ, у яких користувач може інтерактивно маніпулювати 4D-об'єктами. Segerman [6] ілюструє, як тривимірний друк проєкцій 4D-об'єктів дозволяє матеріалізувати їх у спосіб, доступний для безпосереднього тактильного сприйняття – що є альтернативним підходом до проблеми обмеженості екранної візуалізації.

З технічного боку обчислювальна складність 4D-алгоритмів при роботі зі складними многогранниками може бути суттєво знижена шляхом перенесення матричних операцій на GPU за допомогою бібліотеки CuPy – GPU-аналогу NumPy. Для задач реального часу в ігровій індустрії або інтерактивних наукових симуляціях це є необхідною умовою забезпечення прийнятної частоти кадрів навіть для складних 4D-об'єктів із тисячами граней.

Висновки

Мова програмування Python є потужним і зручним інструментом для реалізації алгоритмів 4D-візуалізації завдяки розвиненій екосистемі бібліотек числових обчислень і графіки.

Математичний апарат перспективних проєкцій та матриць повороту у поєднанні з можливостями NumPy і Matplotlib дозволяє ефективно реалізовувати інтерактивну анімацію 4D-об'єктів. Чотиривимірна геометрія знаходить практичне застосування у фізиці, машинному навчанні, медичній візуалізації та ігровій індустрії, що підкреслює актуальність розвитку відповідних програмних інструментів. Подальший прогрес у цій галузі пов'язаний із впровадженням VR/AR-технологій, застосуванням GPU-прискорення та розробкою інтуїтивніших систем керування 4D-поворотами на основі алгебри Кліффорда.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Cavallo M. Higher Dimensional Graphics: Conceiving Worlds in Four Spatial Dimensions and Beyond. Computer Graphics Forum. – 2021. – Vol. 40, No. 2. – P. 51–63. DOI: 10.1111/cgf.142614.
2. Hanson A.J. Visualizing More Quaternions. Elsevier, Amsterdam, 2024. – 516 p. ISBN: 9780323992022.
3. Caplan P.C. Tessellation and interactive visualization of four-dimensional spacetime geometries. arXiv:2403.19036, 2024.
4. Sial A.H., Rashdi S.Y.S., Khan A.H. Comparative analysis of data visualization libraries Matplotlib and Seaborn in Python. International Journal of Advanced Trends in Computer Science and Engineering. – 2021. – Vol. 10, No. 1. – P. 2770–2781. DOI: 10.30534/ijatcse/2021/391012021.
5. Huang H. et al. Towards a comprehensive evaluation of dimension reduction methods for transcriptomic data visualization. Communications Biology. – 2022. – Vol. 5. – P. 719. DOI: 10.1038/s42003-022-03628-x.
6. Igarashi H., Sawada H. 4D Exploring System for Intuitive Understanding of 4D Space. Proceedings of ICAT-EGVE 2023 – International Conference on Artificial Reality and Telexistence / Eurographics Symposium on Virtual Environments. – 2023.
7. Harris C.R. et al. Array programming with NumPy. Nature. – 2020. – Vol. 585. – P. 357–362. DOI: 10.1038/s41586-020-2649-2.
8. Geruganti S. Visualizing the Fourth Dimension: A Comprehensive Framework for 4D Data Representation in Scientific Computing. Engineering Archive. – 2025. DOI: 10.31224/5048.

Присяжнюк Тетяна Іванівна – студентка 4 курсу Вінницького національного технічного університету, факультету інформаційних технологій та комп’ютерної інженерії, групи 6ПІ-22б, Вінниця, e-mail: baconbootyy@gmail.com

Prysiazhniuk Tetiana Ivanivna – 4th year student at Vinnytsia National Technical University, Faculty of Information Technologies and Computer Engineering, group 6PI-22b, Vinnytsia, e-mail: baconbootyy@gmail.com