Преобразование координат объектов с изображения в пикселях в мировые координаты с использованием Python и Unity

Бабчинецкий Сергей Геннадиевич

Гатаулин Александр Денисович

Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, Россия

aleksandrgataulin745@gmail.com

lnpt@guap.ru

Аннотация. В статье приводится описание разработки и реализации алгоритма преобразования координат объектов, зафиксированных на изображении от камеры, установленной на беспилотном летательном аппарате, а также разработки симуляции на движке Unity для проверки написанного алгоритма и верификации полученных мировых координат.

Целью данной работы является разработка и реализация алгоритма преобразования координат в пикселях объектов в мировые географические координаты форматов EPSG3857 (Google Mercator) и EPSG4326 (WGS84), которые являются наиболее распространенными в современном мире. Разрабатываемый алгоритм позволит вычислять географические координаты обнаруживаемых объектов, зная их координаты в пикселях и географические координаты беспилотного летательного аппарата, имеющего датчик GPS для мониторинга его местоположения.

В работе используются математические основы преобразования координат, а также язык программирования Python для выполнения основных математических

операций и игровой движок Unity для симуляции процесса полета беспилотного летательного аппарата и передачи снимков с камеры в Python-скрипт. На движке Unity построена сцена для тестирования алгоритма. С помощью языка C# происходит сбор необходимой для вычисления координат информации и передается на микросервис, запущенный на Flask и обрабатывающий изображения с помощью нейронной сети, которая обучена на задачу детекции дорожных повреждений.

Работа выполнена с использованием известных и простых в изучении библиотек и программных средств, имеющих достаточно учебных пособий для быстрой реализации проекта. Также используются технологии компьютерного зрения и машинного обучения, что подчеркивает актуальность и научную новизну проводимого исследования.

Реализация алгоритма позволяет эффективно обрабатывать данные, получаемые от беспилотных систем, и расширяет возможности их применения в геоинформационных и исследовательских задачах. Разработанный алгоритм может быть использован в различных областях, включая навигацию, картографию, геоинформационные системы, а также в исследованиях, связанных с беспилотными системами и аэрофотосъемкой.

Ключевые слова. Беспилотные летательные аппараты, преобразование координат, географические координаты, EPSG3857, EPSG4326, машинное обучение, Unity, Python, симуляция полета, нейронные сети.

Convert object coordinates from image pixels to world coordinates using Python and Unity

Gataulin Alexander Denisovich, Babchinetsky Sergey Gennadievich

Saint-Petersburg State University of Aerospace Instrumentation, St. Petersburg, Russia

aleksandrgataulin745@gmail.com

lnpt@guap.ru

Annotation. The article describes the development and implementation of an algorithm for transforming the coordinates of objects captured in an image from a camera installed on an unmanned aerial vehicle, as well as the development of a simulation using the Unity engine to test the written algorithm and verify the obtained world coordinates.

The purpose of this work is to develop and implement an algorithm for converting coordinates in pixels of objects into world geographic coordinates of the EPSG3857 (Google Mercator) and EPSG4326 (WGS84) formats, which are the most common in the modern world. The developed algorithm will make it possible to calculate the geographic coordinates of detected objects, knowing their coordinates in pixels and the geographic coordinates of an unmanned aerial vehicle that has a GPS sensor to monitor its location.

The work uses the mathematical foundations of coordinate transformation, as well as the Python programming language to perform basic mathematical operations and the Unity game engine to simulate the flight process of an unmanned aerial vehicle and transfer images from the camera to a Python script. A scene for testing the algorithm was built on the Unity engine. Using the C# language, the information necessary to calculate the coordinates is collected and transferred to a microservice running on Flask and processing images using a neural network that is trained for the task of detecting road damage.

The work was carried out using well-known and easy-to-learn libraries and software that have enough tutorials for quick implementation of the project. Computer vision and machine learning technologies are also used, which emphasizes the relevance and scientific novelty of the research being carried out.

The implementation of the algorithm makes it possible to efficiently process data received from unmanned systems and expands the possibilities of their use in geoinformation and research tasks. The developed algorithm can be used in various fields, including navigation, cartography, geographic information systems, as well as in research related to unmanned systems and aerial photography.

Keywords. Unmanned aerial vehicles, coordinate transformation, geographic coordinates, EPSG3857, EPSG4326, machine learning, Unity, Python, flight simulation, neural networks.

Введение

Современные технологии в области беспилотных летательных аппаратов (БПЛА) предоставляют широкий спектр возможностей в различных сферах, начиная от наблюдения за местностью и заканчивая геоинформационным анализом [1]. В рамках выполнения данных задач является востребованным обнаружение и определение местоположения ключевых объектов. Существуют алгоритмы и модели машинного обучения, включая сверточную нейронную сеть, способные выделить объект на изображении и вычислить его пиксельные координаты [2]. Однако, для полноценного использования данных, полученных с камер, установленных на БПЛА,

необходимо решить задачу преобразования координат объектов на изображении в пикселях в мировые географические координаты.

Объектом исследования в данной статье является процесс преобразования координат объектов, зафиксированных на изображении, полученном от камеры, установленной на беспилотном летательном аппарате (БПЛА). Этот процесс является ключевым моментом при работе с данными, полученными от БПЛА, и представляет собой технологическую задачу, требующую точного преобразования координат в пикселях в мировые географические координаты в формате EPSG3857 (Google Mercator) и EPSG4326 (WGS84) [3].

Авторами статьи обсуждены математические основы данного преобразования и представлено решение данной задачи с использованием языка программирования Python для выполнения основных математических операций [4] и игрового движка Unity [5] для симуляции процесса полета БпЛА и отправки снимков с камеры в Python скрипт. Предметом исследования является разработка данного алгоритма с применение вышеперечисленных инструментов. В процессе выполнения алгоритма производится преобразование координат в пикселях в координаты GPS формата EPSG3857, затем преобразование в формат EPSG4326.

Работа направлена на обеспечение эффективной обработки данных, получаемых от беспилотных систем, и расширения возможностей их применения в геоинформационных и исследовательских задачах.

Исходные данные

Высота аэрофотосъемки с БПЛА [6] рассчитана на диапазон высот от 10 до 50 метров, камера направлена строго вниз, от чего получается изображение с координатами камеры в центре кадра и декартовой системой координат относительно центральной точки.

Для обнаружения целевых объектов на изображении, полученном от БпЛА используется сверточная нейронная сеть модели YoloV8 [7]. При обнаружении объектов результатом обработки изображения является массив bounding боксов [8], представляющих собой координаты x и y левого верхнего и правого нижнего углов рамки. Координаты x и y даны в пикселях с началом отсчета в левом верхнем углу кадра. Поэтому в алгоритме стоит перенести исходную точку в центр изображения.

Переменные параметры:

1. GPS координаты камеры в формате EPSG3857, измеряемой в метрах или EPSG4326, измеряемой в градусах широты и долготы,
2. Высота камеры в метрах (10-50 м),
3. Угол поворота камеры относительно оси Z в градусах (0-360 градусов),
4. Разрешение изображения в пикселях,
5. Угол обзора камеры.

Математическое описание алгоритма

Преобразование пиксельных координат, полученных от модели YoloV8 в координаты GPS EPSG3857 требует выполнения следующих действий:

1. Перенос начальной точки отсчета в центр изображения, относительно которой меняются пиксельные координаты объектов

$$B_x = B_x - \frac{W_{res}}{2}$$

$$B_y = B_y - \frac{H_{res}}{2}$$

Где:

B_x и B_y – координаты точек в по x и y соответственно,

resW, resH – разрешение изображения в пикселях по ширине и высоте.

2. Расчет центров bounding боксов

$$C_x = \frac{B_{x1} + B_{x2}}{2}$$

$$C_y = \frac{B_{y1} + B_{y2}}{2}$$

Где:

C_x и C_y – координаты точек в по x и y соответственно.

3. Нормализация вектора координат x и y

$$\mathrm{C}_x = \frac{2 * \mathrm{C}_x}{W_{res}}$$

$$\mathrm{C}_y = \frac{2 * \mathrm{C}_y}{H_{res}}$$

4. Применение матрицы поворота для выравнивания изображения по северу

5. Расчет координат в метрах относительно положения камеры [9]

$$G_{xy} = \mathrm{C}_{xy} * h_{cam} * \tan\frac{°\mathrm{F}}{2} * \frac{\pi}{180}$$

$$G_x = G_x * \frac{W_{res}}{H_{res}}$$

Где:

G_{xy} – вектор координат объектов, выраженный в метрах,

C_{xy} – нормализованный вектор координат объектов,

°F - угол обзора камеры,

h_{cam} – высота расположения камеры.

Здесь градусное значение внутри функции тангенса преобразуется в радианное и нормализованный вектор Cxy умножается на высоту и тангенс для нахождения вектора координат в метрах. Далее, в связи с растянутостью изображения и искажения перспективы координата в метрах по x умножается на соотношение сторон

6. Расчет координат в метрах в формате EPSG3857

$$G_{xy} = G_{xy} + G_{cam}$$

Где:

G_{cam} – вектор координат камеры, выраженный в метрах.

До этого координаты точек были представлены в системе координат относительно камеры, в данном действии происходит их выражение в метрах относительно Земли, то есть в формате EPSG3857.

Далее эти координаты используются для нахождения широты и долготы в градусах. Используется половина длины экватора для нормализации вектора координат в метрах [10].

Реализация алгоритма на языке Python

Для реализации всех математических и векторных расчетов на языке Python, была использована библиотека Numpy, для работы с изображениями библиотека

Opencv-python и для обнаружения и выделения объектов фреймворк Ultralitycs, предоставляющий новейшую модель сверточной нейронной сети YoloV8.

В рамках задачи было проведено обучение модели на задачу детекции дорожных неровностей, включающих выбоины, проломы, трещины дорожного полотна [11].

Для расчета координат было создано 2 класса-сервиса, проводящих векторные вычисления для преобразования координат в форматы EPSG3857 и EPSG4326 - Pixels2MetresConverter и CRSConverter. Также создается класс для сохранения обработанных изображений для подтверждения корректного обнаружения – ImageSaver. А также класс для обработки полученных фотографий моделью нейронной сети – ModelProcessing.

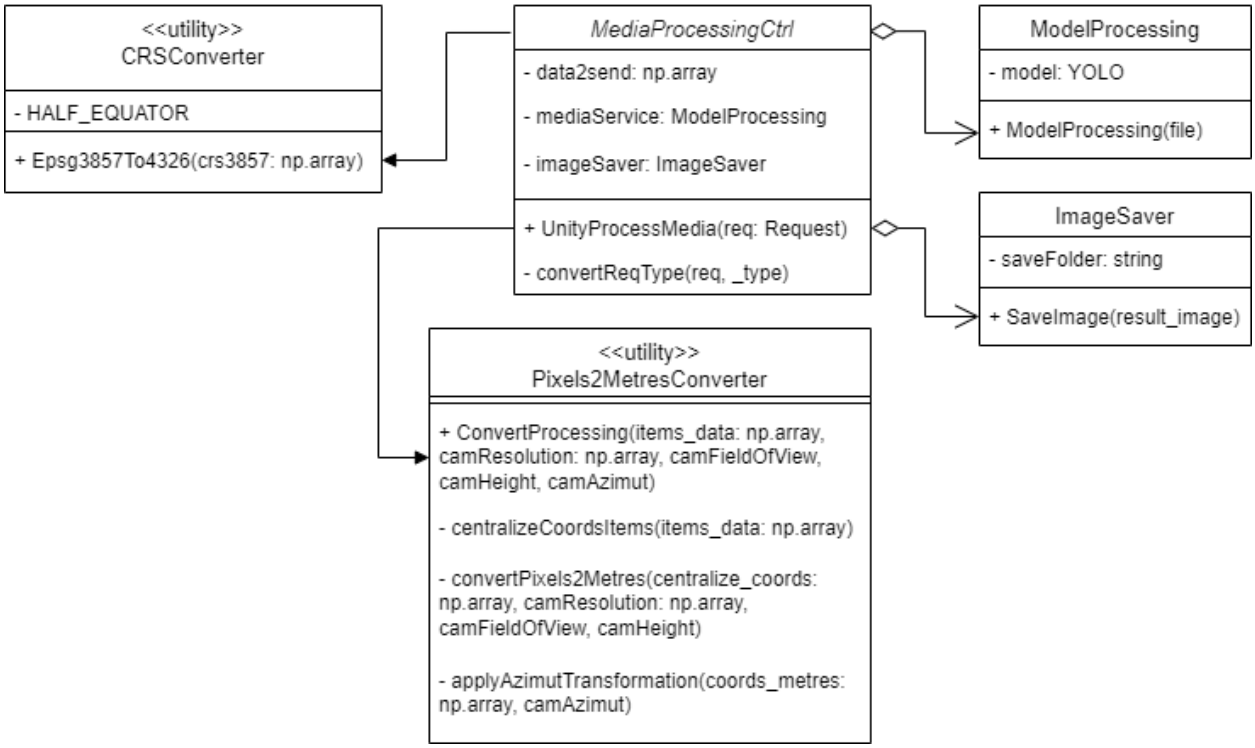

Рис.1 – UML диаграмма классов микросервиса на Flask [12].

С помощью библиотеки Flask через главный файл main.py прослушивается порт и принимает входящие HTTP запросы. По API через URL адрес http//:localhost:5000/

imageProcessing с методом POST вызывается функция process_media(). Далее происходит непосредственный процесс преобразования координат и отправляется ответ в виде массива из пар векторов координат - один в формате EPSG3857, другой в формате EPSG4326. В MediaProcessingCtrl происходит контроль запроса и использование классов-сервисов для грамотной обработки изображения. Между MediaProcessingCtrl и другими классами существует следующая связь:

1. Статический класс Pixels2MetresConverter - связан с помощью ассоциации, контроллер использует только один публичный статический метод для вычисления координат в метрах,
2. Статический класс CRSConverter- связан с помощью ассоциации, контроллер использует только один публичный статический метод для вычисления координат в метрах,
3. ImageSaver связан с контроллером с помощью агрегации, ведь он использует как тип переменной поля класса и вызывает его методы во внутренней логике,
4. ModelProcessing связан с контроллером с помощью агрегации, ведь он использует как тип переменной поля класса и вызывает методы во внутренней логике.

Симуляция процесса аэрофотосъемки на движке Unity

В качестве симуляции процесса проведения аэросъемки был выбран игровой движок Unity. Причины выбора данной платформы:

- Отсутствие возможности проведения аэрофотосъемки с настоящим БпЛА [13],
- Обширные возможности для 3D графики, что позволит с максимальной точностью симулировать сценарии, отражающие реальные условия [14],

- Огромное сообщество и ресурсы, что позволит быстрее ознакомиться с недостающими знаниями,
- Высокий уровень абстракции с использованием языка C#.

Идея заключается в том, чтобы делать скриншот с главной камеры, сохранения его в папке Assets/Screenshots и отправку в запущенный на Flask микросервис по URL адресу http//:localhost:5000/** [15]. Далее принимается ответ в виде массива пар векторов с координатами дорожных неровностей, значения которых сравниваются с местоположением объектов на сцене. Сравниваются координаты в ед. измерения в метрах, то есть в формате EPSG3857, координаты в градусах сравниваются со значениями онлайн преобразователя координат.

Сцена была построена из плоскости, составляющей дорожное полотно. К плоскости прикреплен скрипт для случайного распределения объектов PotholeSpawn.cs. Он создает объекты на сцене на заданной площади, с случайным размером и координатами по x и z. Y координата отвечает за высоту. В объект Canvas добавлены две кнопки, одна отвечает за пересоздание объектов на сцене, вторая за создание скриншота и отправки его на обработку. Над плоскостью расположена камера, направленная на дорожное полотно под прямым углом. С помощью скрипта ScreenshotCapture.cs, присоединенного к объекту MainCamera происходит получение данных о ее местоположении, угле поворота, угле обзора и разрешения. Фиксируется изображение, скриншот сохраняется в папку Assets/Screenshots и отправляется в микросервис, написанный на Flask.

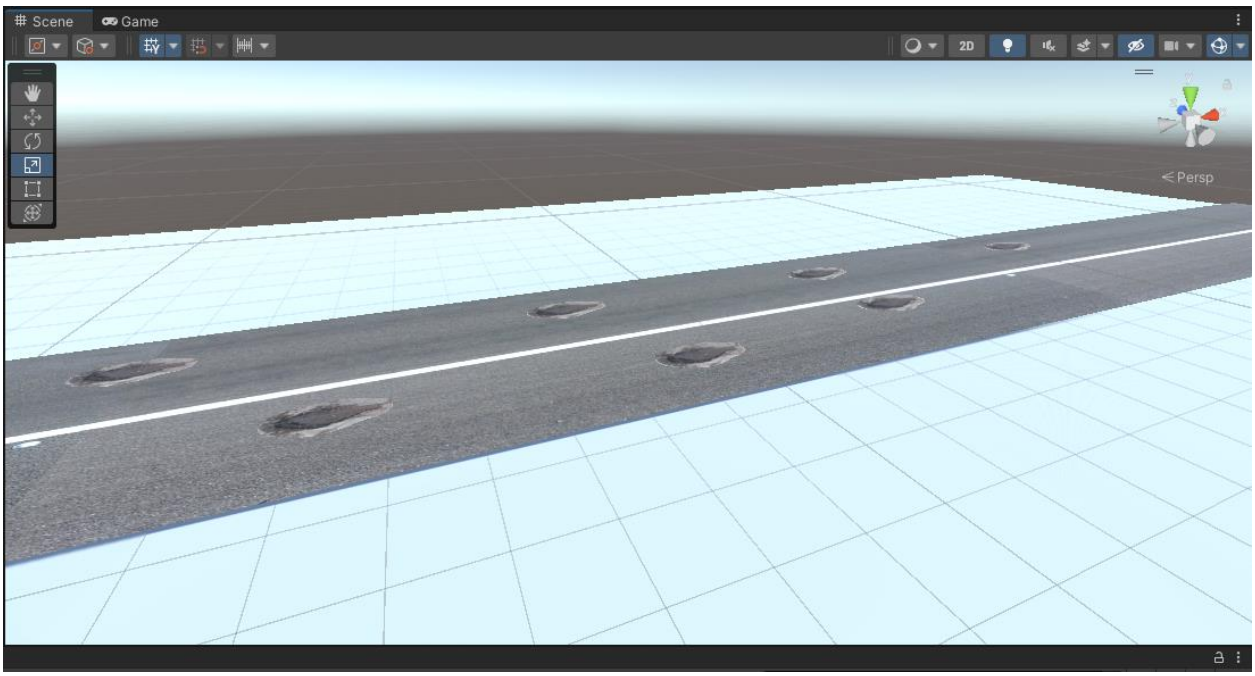

Рис.2 – Сцена с плоскостью дороги и целевыми объектами.

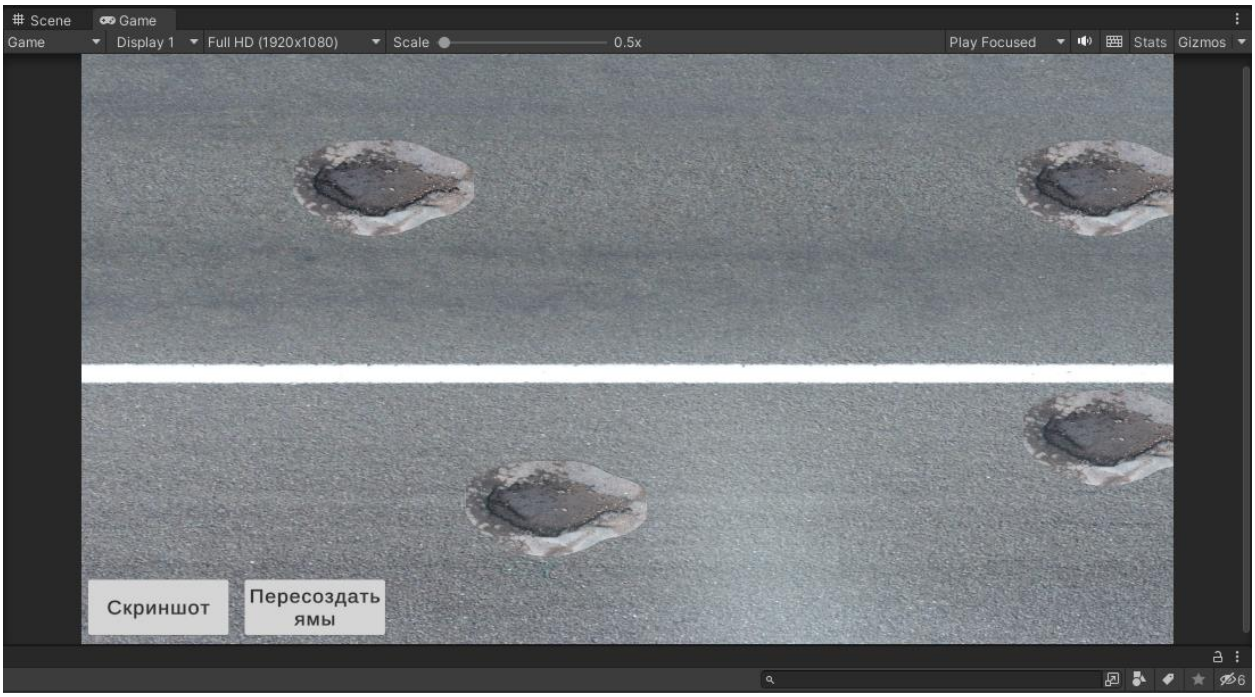

Рис.3 – Интерфейс проведения тестовых испытаний алгоритма.

Передаваемые параметры в форме HTTP запроса [16]:

- **image** – файл изображения с параметром mimetype: «image/png»

- **fieldOfView** – угол обзора камеры в градусах с типом данных float, получаемый из объекта MainCamera с помощью GetComponent<Camera>().fieldOfView
- **cameraHeight** – высота камеры в метрах с типом данных float, получаемая из объекта MainCamera с помощью GetComponent<Transform >().position.y
- **cameraAzimut** – угол поворота камеры в градусах с типом данных float, получаемый из объекта MainCamera с помощью GetComponent<Transform >().rotation.y
- **cameraX** и **cameraY** – координаты позиции камеры в метрах с типом данных float, выраженные в формате EPSG3857, получаемые из объекта MainCamera с помощью GetComponent<Transform>().position.x и GetComponent<Transform>().position.z соответственно.
- **screenWidth** и **screenHeight** – разрешение кадра в пикселях с типом данных integer, полученных при обращении к статическому классу UnityEngine.Screen к полям width и height.

Полученный массив пар координат выводится в консоль и сверяется с реальными данными.

Результаты тестов

По результатам тестов удалось добиться точности измерений с погрешностью 0.5 метров. Координаты в метрах, пришедшие из скрипта Python практически идеально совпадают с координатами ям на сцене. А координаты широты и долготы, выраженные в формате EPSG4326, сверенные с онлайн преобразователем координат, совпадают с точностью ±0.00005 градусов.

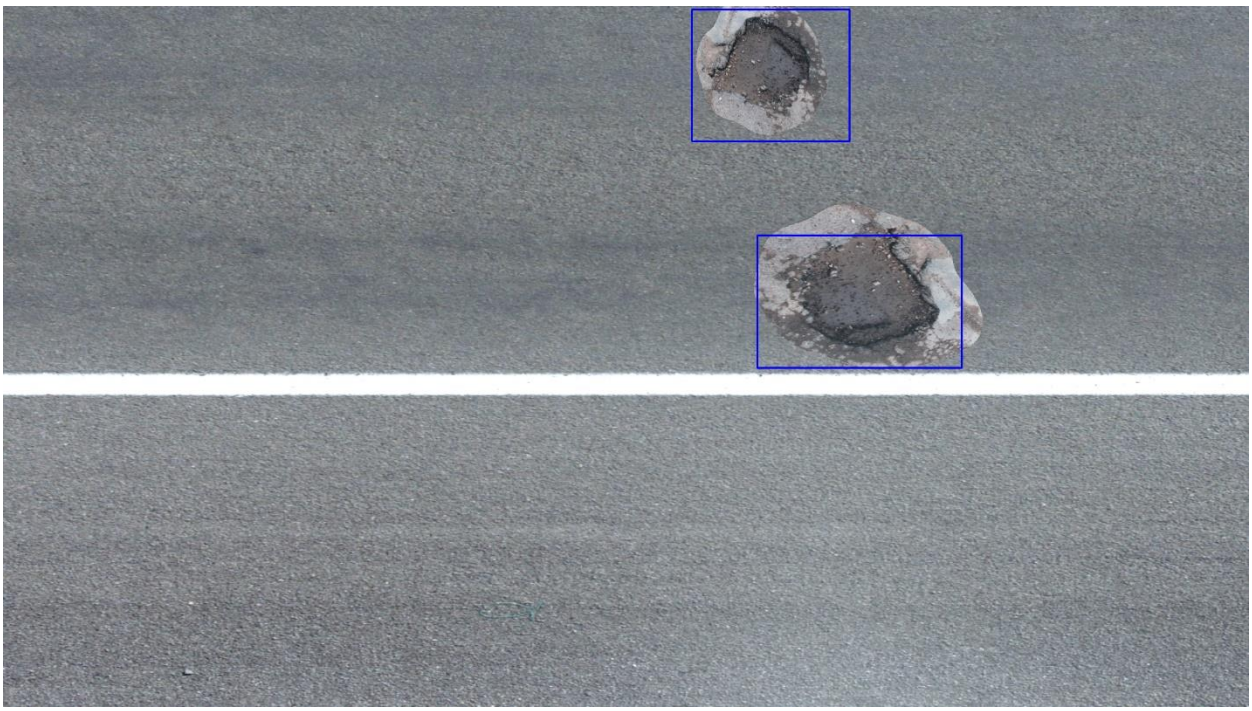

Рис. 4 – Детектированные дорожные повреждения моделью YoloV8 [17].

Как видно на рис. 4, в bounding box обведены две выбоины. Для проверки точности алгоритма был взят объект, находящийся чуть ниже другого. Координаты, полученные при математических расчетах равны:

EPSG3857: $X_1 = 16.38958$, $Y_1 = 3.880542$

EPSG4326: $Lon_1 = 0.000147230140117$, $Lat_1 = 0.000034859503927$

Lat и Lon – это широта и долгота соответственно. При этом реальные данные, взятые из сцены Unity и онлайн преобразователя систем координат соответствуют значениям:

EPSG3857: $X_2 = 17.26886$, $Y_2 = 4.604198$

EPSG4326: $Lon_2 = 0.000155128808773$, $Lat_2 = 0.000041360214341$

Итоговые погрешности в метрах и градусах в данном испытании составила:

$$\sqrt{|X_2 - X_1|^2 + |Y_2 - Y_1|^2} = 1{,}2968 \text{ м}$$

$$\sqrt{|Lon_2 - Lon_1|^2 + |Lat_2 - Lat_1|^2} = 0{,}00001022977\ ^\circ$$

Заключение

В ходе исследования и разработки алгоритма преобразования координат объектов, полученных от камеры на беспилотном летательном аппарате (БПЛА), в мировые географические координаты [18], достигнуты значительные результаты. Тестирование алгоритма продемонстрировало высокую точность преобразования координат с погрешностью всего 0.5 метров.

Координаты в метрах, полученные из Python-скрипта, практически идеально совпадают с реальными координатами объектов на сцене Unity. Точность алгоритма подтверждается результатами сравнения координат широты и долготы в формате EPSG4326 с онлайн преобразователем координат, где наблюдается совпадение с точностью ±0.00005 градусов.

На основании анализа bounding box [19], обведенного вокруг двух выбоин на изображении, видно, что координаты, полученные из математических расчетов, близки к реальным данным в пределах погрешности. При этом итоговые погрешности в метрах и градусах составили 1.2968 м и 0.00001022977 ° соответственно.

Эти результаты подтверждают эффективность разработанного алгоритма в преобразовании координат, что делает его ценным инструментом для обработки данных, полученных от беспилотных систем. Полученные выводы подчеркивают потенциал применения данного метода в геоинформационных и исследовательских задачах, обеспечивая высокую точность и надежность преобразования координат объектов на изображении.

С учетом того, что данная технология позволяет оперировать с изображениями, полученными от камеры, установленной на беспилотном летательном аппарате, она

может быть применена в космических миссиях для наблюдения и анализа поверхности планет, астероидов и других космических объектов. Это может быть особенно полезно для картографирования и изучения местности перед высадкой роботов-исследователей или для мониторинга космических объектов и обнаружения потенциальных опасностей.

На данный момент есть некоторые способы преобразовывать координаты объекта на изображении в пикселях в координаты, выраженные в метрах на языке C++ с применением библиотеки opencv [20]. Преимуществами системы, представленной в данной статье является использование языка программирования Python, учет угла поворота камеры и дальнейшее преобразования из системы координат EPSG3857 в формат EPSG4326.

Библиографический список

1. Вопросы оценки выполнимости задач, поставленных перед беспилотными летательными аппаратами / Н. Г. Джавадов, Ф. Г. Агаев, Г. А. Гусейнов, П. Р. г. Зульфугарлы // Труды МАИ. – 2022. – № 127. – DOI 10.34759/trd-2022-127-20.
2. Олькина, Д. С. Алгоритм семантической сегментации изображений для решения задачи позиционирования летательного аппарата на земной поверхности / Д. С. Олькина // Труды МАИ. – 2023. – № 130. – DOI 10.34759/trd-2023-130-18.
3. Gumelar O. et al. Remote sensing image transformation with cosine and wavelet method for SPACeMAP Visualization //IOP Conference Series: Earth and Environmental Science. – IOP Publishing, 2020. – Т. 500. – №. 1. – pp. 012079.

4. Harris C. R. et al. Array programming with NumPy //Nature. – 2020. – Т. 585. – №. 7825. – pp. 357-362.
5. Juliani A. et al. Unity: A general platform for intelligent agents //arXiv preprint arXiv:1809.02627. – 2018.
6. В. А. Ненашев, В. И. Афанасьева, А. А. Залищук [и др.]. Формирование трехмерных моделей местности на основе лидарной съемки для выявления структурных изменений земной поверхности // Труды МАИ. – 2023. – № 131. – DOI 10.34759/trd-2023-131-15.
7. А. М. Преснецов, А. П. Тюрин. Разработка программно-аппаратного комплекса для мониторинга производственной деятельности с использованием нейросети YOLOv8 // Интеллектуальные системы в производстве. – 2023. – Т. 21, № 2. – С. 140-151. – DOI 10.22213/2410-9304-2023-2-140-151.
8. He Y. et al. Bounding box regression with uncertainty for accurate object detection //Proceedings of the ieee/cvf conference on computer vision and pattern recognition. – 2019. – pp. 2888-2897.
9. Г. А. Калиниченко, С. В. Скороход. Анализ преобразования изображения из локальной системы координат в систему координат камеры // Actual scientific research 2018 : материалы XXXVII Международной научно-практической конференции, Москва, 27 апреля 2018 года. – Москва: Научный центр "Олимп", 2018. – С. 143-145.
10. Czogalla O., Naumann S. Pedestrian guidance for public transport users in indoor stations using smartphones //2015 IEEE 18th International Conference on Intelligent Transportation Systems. – IEEE, 2015. – pp. 2539-2544.

11. Полянцева К. А. Высоконагруженная платформа для агрегации и анализа неструктурированных данных о состоянии дорожного полотна // Автоматизация в промышленности, Учредители: Издательский дом" ИнфоАвтоматизация". – №. 5. – С. 32-37.
12. I. Zaretska, O. Kulankhina, H. Mykhailenko, T. Butenko. Consistency of UML Design // International Journal of Information Technology and Computer Science. – 2018. – Vol. 10, No. 9. – pp. 47-56. – DOI 10.5815/ijitcs.2018.09.06.
13. Е. А. Наугольных, И. Л. Бартоломей. Аэрофотосъемка с БПЛА для обследования искусственных сооружений на автомобильных дорогах // Модернизация и научные исследования в транспортном комплексе. – 2022. – Т. 1. – С. 323-325.
14. К. Ю. Жигалов. Использование игровых визуализаторов графики в современных геоинформационных системах // Cloud of Science. – 2016. – Т. 3, № 1. – С. 71-80.
15. Vuksanovic I. P., Sudarevic B. Use of web application frameworks in the development of small applications //2011 Proceedings of the 34th International Convention MIPRO. – IEEE, 2011. – pp. 458-462.
16. Кузнецова, С. В. Особенности кросс-платформенной разработки мобильных приложений с использованием Xamarin / С. В. Кузнецова // Труды МАИ. – 2022. – № 125. – DOI 10.34759/trd-2022-125-21.
17. H. H. Ngo. Vehicle-detection-based traffic density estimation at road intersections // International Journal of Open Information Technologies. – 2023. – Vol. 11, No. 7. – pp. 39-46.

18. B. K. Choi, J. Uk. Park, K. Min Roh, S. J. Lee. Comparison of GPS receiver DCB estimation methods using a GPS network // Earth, Planets and Space. – 2013. – Vol. 65, No. 7. – pp. 707-711. – DOI 10.5047/eps.2012.10.003.
19. N. Ravi, S. Naqvi, M. El-Sharkawy. BIoU: An Improved Bounding Box Regression for Object Detection // Journal of Low Power Electronics and Applications. – 2022. – Vol. 12, No. 4. – pp. 51. – DOI 10.3390/jlpea12040051.
20. Opencv получить координаты мировой системы координат из координат пикселей, дата обращения: 01.03.2024. URL: https://russianblogs.com/article/9367131104/#_11

References

1. Questions of assessing the feasibility of tasks assigned to unmanned aerial vehicles / N. G. Javadov, F. G. Agaev, G. A. Guseinov, P. R. Zulfugarly // Proceedings of MAI. – 2022. – No. 127. – DOI 10.34759/trd-2022-127-20.
2. Olkina, D. S. Semantic image segmentation algorithm for solving the problem of positioning an aircraft on the earth's surface / D. S. Olkina // Proceedings of MAI. – 2023. – No. 130. – DOI 10.34759/trd-2023-130-18.
3. Gumelar O. et al. Remote sensing image transformation with cosine and wavelet method for SPACeMAP Visualization //IOP Conference Series: Earth and Environmental Science. – IOP Publishing, 2020. – T. 500. – No. 1. – pp. 012079.
4. Harris C. R. et al. Array programming with NumPy //Nature. – 2020. – T. 585. – No. 7825. – pp. 357-362.
5. Juliani A. et al. Unity: A general platform for intelligent agents //arXiv preprint arXiv:1809.02627. – 2018.

6. V. A. Nenashev, V. I. Afanasyeva, A. A. Zalischuk [and others]. Formation of three-dimensional terrain models based on lidar survey to identify structural changes in the earth's surface // Proceedings of MAI. – 2023. – No. 131. – DOI 10.34759/trd-2023-131-15.
7. A. M. Presnetsov, A. P. Tyurin. Development of a software and hardware complex for monitoring production activities using the YOLOv8 neural network // Intelligent systems in production. – 2023. – T. 21, No. 2. – pp. 140-151. – DOI 10.22213/2410-9304-2023-2-140-151.
8. He Y. et al. Bounding box regression with uncertainty for accurate object detection //Proceedings of the ieee/cvf conference on computer vision and pattern recognition. – 2019. – pp. 2888-2897.
9. G. A. Kalinichenko, S. V. Skorokhod. Analysis of image transformation from a local coordinate system to a camera coordinate system // Actual scientific research 2018: materials of the XXXVII International Scientific and Practical Conference, Moscow, April 27, 2018. – Moscow: Scientific Center “Olympus”, 2018. – pp. 143-145.
10. Czogalla O., Naumann S. Pedestrian guidance for public transport users in indoor stations using smartphones //2015 IEEE 18th International Conference on Intelligent Transportation Systems. – IEEE, 2015. – pp. 2539-2544.
11. Polyantseva K. A. High-load platform for aggregation and analysis of unstructured data on the condition of the roadway // Automation in industry, Founders: Publishing house "InfoAutomation". – No. 5. – pp. 32-37.

12. I. Zaretska, O. Kulankhina, H. Mykhailenko, T. Butenko. Consistency of UML Design // International Journal of Information Technology and Computer Science. – 2018. – Vol. 10, No. 9. – pp. 47-56. – DOI 10.5815/ijitcs.2018.09.06.

13. E. A. Naugolnykh, I. L. Bartolomei. Aerial photography from a UAV for the inspection of artificial structures on highways // Modernization and scientific research in the transport complex. – 2022. – T. 1. – pp. 323-325.

14. K. Yu. Zhigalov. Using game graphics visualizers in modern geographic information systems // Cloud of Science. – 2016. – T. 3, No. 1. – pp. 71-80.

15. Vuksanovic I. P., Sudarevic B. Use of web application frameworks in the development of small applications //2011 Proceedings of the 34th International Convention MIPRO. – IEEE, 2011. – pp. 458-462.

16. Kuznetsova, S.V. Features of cross-platform development of mobile applications using Xamarin / S.V. Kuznetsova // Proceedings of MAI. – 2022. – No. 125. – DOI 10.34759/trd-2022-125-21.

17. H. H. Ngo. Vehicle-detection-based traffic density estimation at road intersections // International Journal of Open Information Technologies. – 2023. – Vol. 11, No. 7. – pp. 39-46.

18. B. K. Choi, J. Uk. Park, K. Min Roh, S. J. Lee. Comparison of GPS receiver DCB estimation methods using a GPS network // Earth, Planets and Space. – 2013. – Vol. 65, No. 7. – pp. 707-711. – DOI 10.5047/eps.2012.10.003.

19. N. Ravi, S. Naqvi, M. El-Sharkawy. BIoU: An Improved Bounding Box Regression for Object Detection // Journal of Low Power Electronics and Applications. – 2022. – Vol. 12, No. 4. – pp. 51. – DOI 10.3390/jlpea12040051.

20. Opencv get the coordinates of the world coordinate system from pixel coordinates, accessed: 03/01/2024. URL: https://russianblogs.com/article/9367131104/#_11