УДК 004.896

ПРОГРАММНЫЙ ИНСТРУМЕНТ ДЛЯ СОЗДАНИЯ 3D-КАРТ В GAZEBO НА ОСНОВЕ ПРОИЗВОЛЬНЫХ ИЗОБРАЖЕНИЙ И ДАННЫХ ЛАЗЕРНОГО СКАНИРОВАНИЯ

А.А. Закиев (*zaufar@gmail.com*)
Р.О. Лавренов (*lavrenov@it.kfu.ru*)
Е.А. Магид (*magid@it.kfu.ru*)
Высшая Школа Информационных Технологий и Систем,
Казанский Федеральный Университет, Казань

Аннотация. Разработка алгоритмов для мобильных роботов, осуществляющих навигацию, построение карт, в том числе, одновременно с локализацией, требует должного моделирования окружающей среды. Наша статья посвящена программному инструменту, который позволяет автоматически создавать реалистичные 3D-ландшафты, основанные на реальных сенсорных данных. Утилита предусматривает фильтрацию карты занятости на входе и импорт её в систему Gazebo в качестве карты высот с возможностью настройки создаваемой в симуляции среды.[1]

Ключевые слова: Gazebo; ROS; октокарта; решетка занятости; карта высот; фильтрация карты.

Введение

Разработка и развитие мобильных робототехнических систем идут очень интенсивно в сегодняшнее время. Проблема навигации – одна из основных трудностей, встающих перед мобильными роботами. Важными частями задачи навигации автономного устройства являются локализация, построение карты и планирование маршрута [Fuentes-Pacheco et al., 2015]. Локализация отвечает за точное определение нынешней позиции робота в окружающем пространстве. Построение карты происходит посредством

[1] Работа поддержана Российским фондом фундаментальных исследований (РФФИ) и Министерством науки и технологий и Государством Израиль (совместный проект ID 15-57-06010). Часть работ выполнена в соответствии с Государственной программой конкурентоспособности Российской Федерации при Казанском федеральном университете.

сбора сенсорных данных робота и сохранением их в удобной для последующей обработки форме. Используя данные, полученные в ходе построения карты, становится возможным планирование маршрута, позволяющего из стартовой позиции достигнуть цели, не допуская столкновения с препятствиями. Метод одновременной локализации и построения карты (SLAM) объединяет два процесса для более эффективного обеспечения автономной навигации. Алгоритмы SLAM способны справляться с различными ограничениями аппаратного или внешнего характера, например, SLAM методы для монокулярных систем [A. Buyval et al., 2016], SLAM методы, использующие одновременно лидары и визуальные сенсоры [R. W. Wolcott et al., 2014].

Навигационные алгоритмы должны быть тщательно проверены перед их интеграцией в ПО реального робота. Тестирование обычно проводится с помощью компьютерных симуляций, которые являются эффективным и недорогим способом убедиться в работоспособности новых методов, корректности алгоритмов и возможности их применения в существующих робототехнических системах [I. Afanasyev et al., 2015]. Лучший способ создать окружающую среду в симуляции – это использовать реальные сенсорные данные. Для нашего исследования SLAM применительно к гетерогенным группам роботов, использующих совместно создаваемую карту и многоагентное планирование маршрута, мы используем Robot Operating System (ROS). Однако, до сих пор не существовало простого и удобного инструмента для обработки и импорта реальных сенсорных данных, и разработка утилиты с таким функционалом является основным вкладом нашей работы.

Статья организована по следующему принципу. Глава 1 формулирует цели проекта и начальные условия симуляции. В главе 2 дан обзор фильтров и дано обоснование выбора для реализации. Глава 3 содержит в себе детали процесса импорта карты в симуляцию. Глава 4 демонстрирует дополнительные возможности утилиты. Далее подводятся итоги работы.

1. Цель и настройки системы

1.1 Цель исследования

Основная цель нашего исследования связана с изучением взаимодействия гетерогенных групп роботов, состоящих из наземных и малых воздушных автономных устройств с особым вниманием к работе в изменчивой среде. Окружающая среда может быть представлена в виде неточной, неполной или неактуальной картой. Для обеспечения надежной автономной навигации роботы совместно изучают окружающую среду и планируют действия, принимая в расчет различные источники

неопределенности. Мы достигли прогресса в SLAM и планировании пути в совместно создаваемой карте [V. Indelman et al., 2015], и продолжим изучать подходы, которые позволят учитывать изменчивость среды. В частности, такой фреймворк позволит роботам правильно действовать, улучшая локализацию и качество создаваемой карты даже в условиях недостаточности GPS сигнала.

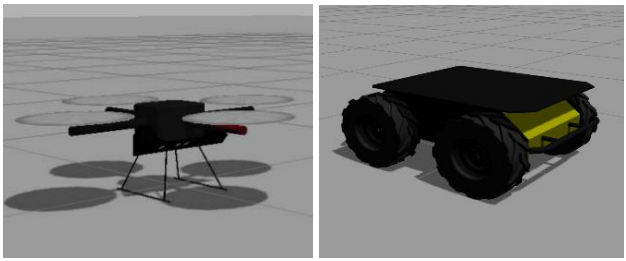

Рис. 1. Квадрокоптеры стандартной конструкции с сенсором Kinect (слева), модель робота Husky в Gazebo (справа)

Наша первая и очень упрощенная модель взаимодействия состояла из двух квадрокоптеров, строивших карту на основе данных с сенсоров Kinect, и робота Husky, планировавшего свой маршрут [M. Sokolov et al., 2016] (Рис. 1). Husky использовал данные с квадрокоптеров и свой лидар для создания графа Вороного, строил основной маршрут и следовал ему, внося локальные корректировки для избегания столкновений с препятствиями. Симуляция проводилась в искусственной среде ROS/Gazebo и все фазы работы роботов (3D-картографирование, вычисление пути и передвижение соответственно) очень сильно зависели от изменений окружающего ландшафта. Чтобы усовершенствовать симуляцию, для создания ландшафта мы хотим использовать карту, основанную на реальных сенсорных данных.

1.2 Условия симуляции

В нашем исследовании мы используем ROS Indigo (Robot Operating System) вместе с симулятором Gazebo 2.2, в котором имеется встроенный физический движок, удобные программные и графический интерфейсы, что позволяет создавать высококачественные симуляции. Мы осуществляем картографирование, используя модели решеток занятости и октокарт в средах ROS/Gazebo.

Решетка занятости является сеткой значений, каждое из которых обозначает наличие препятствия в указанной области [Open Source Robotics

Foundation, 2017]. Значения могут бинарными (0 для пустых ячеек, 1 для занятых препятствием ячеек) или принимать значения в заданном диапазоне, обозначая неровности поверхности и проходимость указанной области [S. Singh et al., 2000]. Когда значения занятости варьируются от 0 до 255, решетка занятости может быть визуализирована в виде изображения в оттенках серого с значениями равными 0 и 255 для полностью свободных и абсолютно непроходимых областей соответственно (рис. 3).

Другим способом хранить данные о занятости 3D-пространства является использование октокарт. Для эффективного хранения информации весь объем пространства делится на вокселы (англ. voxels) и состояние каждого воксела сохраняется в узлах соответствующего октодерева (дерева с коэффициентом ветвления равным 8) [Hornung et al., 2013]. Это позволяет управлять разрешением карты путем ограничения глубины октодерева. Пример деления объема на вокселы и соответствующее делению октодерево показаны на рис. 2.

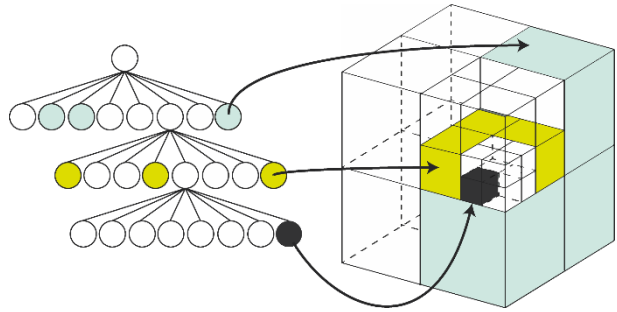

Рис. 2. Пример деления объема на вокселы и соответствующее делению октодерево

1.3 Входные данные на основе реальных сенсорных данных

Чтобы создать реалистичную симуляцию и удостовериться в применимости предложенного решения мы использовали карту реального помещения. Данные были получены в результате лазерного сканирования в Лаборатории Автономного движения и Восприятия в Технионе – Израильском Институте Технологии (рис.3, слева). Вторая карта (рис.3, справа) была получена с сайта Hackaday (https://hackaday.com/tag/lidar/). Карты были создана с помощью ROS-пакета *gmapping* и могут быть сохранены с помощью ROS-пакета *map_server*. Пиксели могут принимать три возможных значения: «черный» для занятых областей, «белый» для пустого пространства и «серый» для тех областей, для которых данные

сканирования отсутствуют. Для поддержки функционала планирования маршрута карты должна быть отфильтрованы перед импортом в Gazebo.

2. Фильтрация карты

Существующие фильтры для уменьшения шумов на изображениях можно с определенной долей условности разделить на следующие типы: линейные, нелинейные и фильтры с нечеткой логикой [Nachtegael, M, et al., 2001]; отдельно стоит упомянуть salt-and-pepper фильтры [Toh et al., 2010].

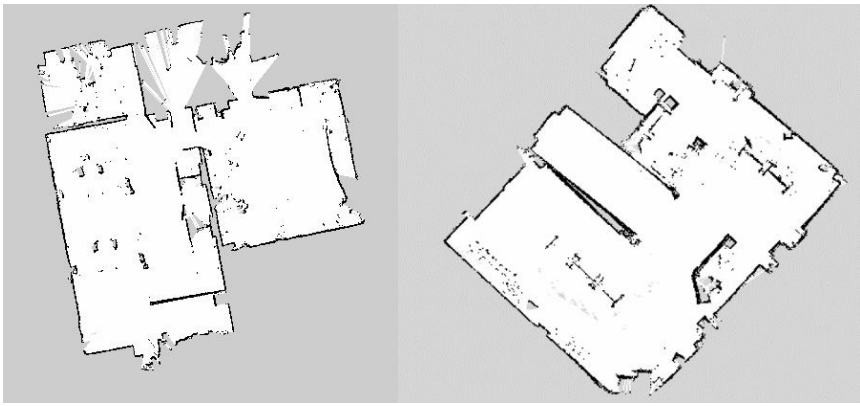

Рис. 3. Исходные карты на основе сенсорных данных

2.1 Выбор фильтра для реализации

На исходной карте пиксели помечены как «белые» (свободные области), «черные» (препятствия) или «серые» (области, не попавшие в зону обзора сенсоров). Это означает, что зашумленные пиксели имеют точно такие же значения, что и незашумлённые. Более того, все шумы являются импульсными, т.е. они не неравномерно распределены по карте.

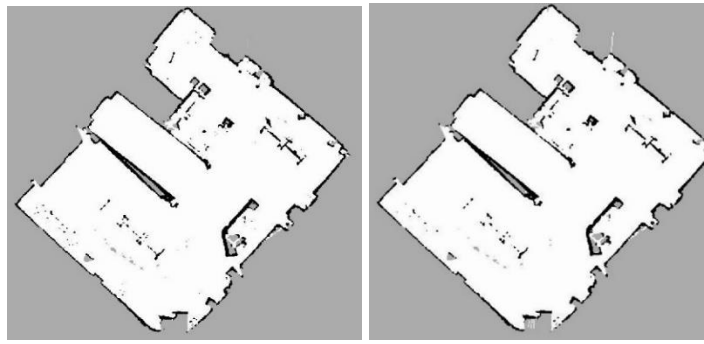

Рис. 4. Исходная карта на основе сенсорных данных, отфильтрованная стандартным (слева) и модифицированным (справа) медианным фильтром

Наличие только трех возможных значений пикселей делает невозможным использование всех фильтров, которые при фильтрации генерируют средние значения, т.к. эти значения не могут быть интерпретированы в терминах занятости или незанятости соответствующей области на карте. Фильтры с нечеткой логикой потенциально могут показать себя лучше, но их применение требует значительных усилий для адаптации алгоритмов к конкретной карте. Salt-and-pepper фильтры могут показаться лучшим выбором для устранения импульсных шумов, но их слабость – это этап определения зашумленных пикселей, т.к. в нашем случае невозможно отличить зашумленный пиксель от незашумленных в силу отсутствия разницы в их значениях.

Поэтому мы используем нелинейный медианный фильтр, который не генерирует средние значения, эффективно устраняет импульсные шумы и прост в реализации. Фильтр реализован на языке C++ и оформлен в виде ROS-пакета. На вход принимается решетка занятости, на выход же подается отфильтрованная решетка занятости в том же формате. На рис. 4 показана схема взаимодействия фильтра с другими ROS-узлами. Все фазы – обработка исходной карты, ее фильтрация и сохранение были объединены в одну утилиту со множеством настраиваемых параметров. Такое объединение делает фильтрацию легким и удобным. Пример отфильтрованной карты показан на рис. 4 (слева) и на рис. 5 (посередине).

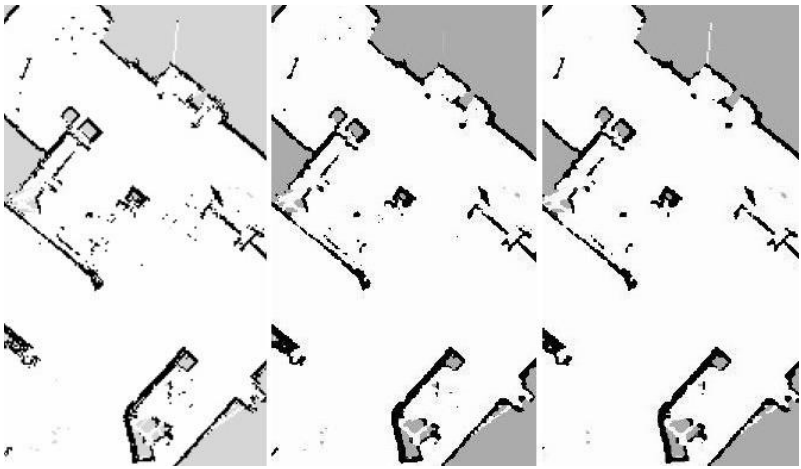

Рис. 5. Увеличенные области исходной карты (слева), отфильтрованной медианным фильтром (в центре) и отфильтрованной модифицированной медианным фильтром (справа)

Однако мы заметили, что тонкие структуры вроде внутренних перегородок исчезли в ходе фильтрации. Чтобы устранить этот недостаток, мы модифицировали медианный фильтр. Его улучшенная версия не воздействует на пиксели, ближайшие соседи которых по строке или столбцу имеют то же значение, что и сам фильтруемый пиксель. Эти соседи в принятой нами нотации будут обозначаться как *(i,j-1), (i,j+1)* в случае строки и *(i-1,j), (i+1,j)* в случае столбца. Результат работы модифицированного фильтра показан на рис. 4 (справа) и на рис. 5 (справа).

3. Импорт карты

Мы создали ландшафт, используя SDF-элемент *heightmap*, которому требуется файл изображения в качестве базы для создания рельефа. Также этот элемент позволяет легко контролировать размеры и максимальную высоту создаваемой карты высот. Попытка использовать исходную карту без обработки была неудачной, т.к. высоты в созданном ландшафте были инвертированы (на левом изображении рис. 6 можно увидеть данный эффект). Такое поведение симулятора вызвано требованиями к базовому изображению элемента *heightmap*: изображение должно быть формате PNG и в режиме оттенков серого, вместо RGB-режима, использующегося по умолчанию.

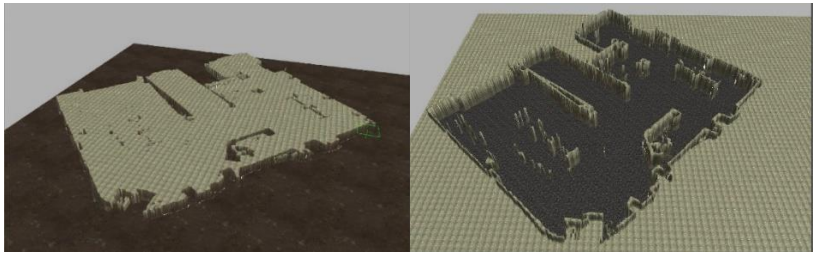

Рис. 6. Результат импорта изображения в RGB-режиме: инвертированные высоты (слева) и успешно импортированная карта высот (справа)

Наша утилита решает эту проблему, проводя все необходимые преобразования в автоматическом режиме. Преимуществом здесь является возможность использовать в качестве базового изображение любого формата. На выходе утилита предоставляет отфильтрованную, отредактированную и полностью готовую к использованию в симуляции карту. Карта высот на основе такого изображения поддерживает обработку столкновений, наложение текстур, обработку света и теней. На правом изображении рис. 6 демонстрируется успешно созданная окружающая среда на основе исходной карты (рис .3 справа).

3.1 Проверка работоспособности симуляции

Чтобы проверить работоспособность и качество созданного утилитой ландшафта, мы провели решение навигационных задач с упомянутыми выше роботами Husky и двумя квадрокоптерами. Во время тестирования производительность симуляции отслеживалась с помощью показателя Real Time Factor (RTF), который отображает отношение времени, прошедшего в симуляции ко времени, которое понадобилось для обработки этого периода времени. Например, если обработка 1 секунды работы симуляции занимает 4 секунды реального времени, то RTF равен 0.25. Таким образом, чем выше RTF, тем эффективнее симуляция.

Несколько моделей роботов были использованы для проверки. Для двух передвигающихся квадрокоптеров RTF равнялся 0.7. Для двух квадрокоптеров с роботом Husky показатель RTF падал до неприемлемого значения 0.01; при использовании с двумя квадрокоптерами робота TurtleBot [M. W. Tully Foote, 2017] значение RTF снижалось до приемлемого для комфортной работы 0.3. Это подтвердило наше предположение о сложной обработке взаимодействия между картой высот и роботом Husky. На рис. 7 показана симуляция с роботом TurtleBot и двумя квадрокоптерами, выполняющими навигационные задачи на созданном по отфильтрованной карте (рис. 4 правое изображение) ландшафте.

Рис. 7. Финальная карта с роботами TurtleBot и двумя квадрокоптерами, выполняющими навигационные задачи

4. Возможности утилиты

После запуска исходная карта обрезается (или расширяется), ее режим отображения меняется на режим оттенков серого. Далее идут этапы фильтрации и инверсии цветов, если они указаны пользователем. После этого карта сохраняется в формате PNG. Последним шагом является создание world-файла для Gazebo и запуск симулятора. Для удобства вся утилита интегрирована с ROS.

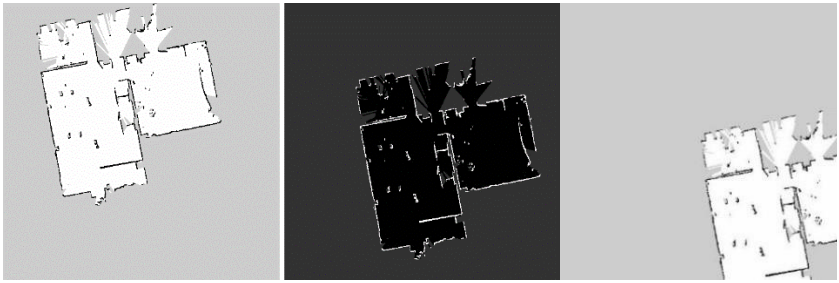

Рис. 8. Результаты запусков утилиты с разными параметрами

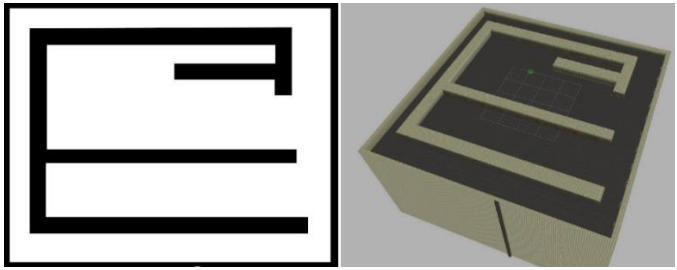

Рис. 9. Пример работы утилиты – из исходного изображения (слева) в среду Gazebo (справа)

Заключение

В этой статье мы представили разработку утилиты для автоматического создания реалистичных ландшафтов в среде симулятора Gazebo на основе изображений (рис. 9) и карт, созданных в ходе сенсорных исследований реального мира. Утилита позволяет проводить фильтрацию и полную подготовку изображения к импорту в качестве карты высот в Gazebo в автоматическом режиме.

В дальнейшем мы планируем проверить работоспособность представленной утилиты, используя модели роботов, имеющиеся в ROS, в том числе и модель робота Инженер [M. Sokolov et al., 2016]. Утилита будет дорабатываться и будет в свободном доступе в качестве ROS-пакета.

Список литературы

[Fuentes-Pacheco et al., 2015] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. Artificial Intelligence Review, November 2015.

[A. Buyval et al., 2016] A. Buyval, I. Afanasyev, and E. Magid, Comparative analysis of ROS-based Monocular SLAM methods for indoor navigation, The 9th Int. Conf. on Machine Vision, 2016.

[R. W. Wolcott et al., 2014] R. W. Wolcott, R. M. Eustice. Visual Localization within LIDAR Maps for Automated Urban Driving, 2014.

[I. Afanasyev et al., 2015] I. Afanasyev, A. Sagitov, and E. Magid. ROS-Based SLAM for a Gazebo-Simulated Mobile Robot in Image-Based 3D Model of Indoor Environment. Advanced Concepts for Intelligent Vision Systems, Springer International Publishing, pp.273-283, 2015.

[V. Indelman et al., 2015] V. Indelman, L. Carlone and F. Dellaert, ""Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments.", The Int. Journal of Robotics Research, vol. 34, no. 7, pp. 849-882, 2015.

[M. Sokolov et al., 2016] M. Sokolov, R. Lavrenov, A. Gabdullin, I. Afanasyev and E. Magid. 3D modelling and simulation of a crawler robot in ROS/Gazebo. The 4th Int. Conf. on Control, Mechatronics and Automation, 2016.

[Open Source Robotics Foundation, 2017] Open Source Robotics Foundation, "nav_msgs/OccupancyGrid Documentation"

[S. Singh et al., 2000] S. Singh, R. Simmons, T. Smith, A. Stentz, V. Verma, A. Yahja & K. Schwehr. Recent Progress in Local and Global Traversability for Planetary Rovers. IEEE Int. Conf. on Robotics and Automation, 2000.

[Hornung et al., 2013] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, *34*(3), 189-206.

[S. K. Johannes Meyer, 2014] S. K. Johannes Meyer, "hector_quadrotor - ROS Wiki," http://wiki.ros.org/hector_quadrotor, 2014.

[Clearpath Robotics, 2015] Clearpath Robotics, "Robots/Husky - ROS Wiki," http://wiki.ros.org/Robots/Husky, 2015

[Nachtegael, M, et al., 2001] Nachtegael, M., Van der Weken, D., Van De Ville, D., Kerre, E., Philips, W., & Lemahieu, I. (2001). An overview of classical and fuzzy-classical filters for noise reduction. *The 10th IEEE Int. Conf. on Fuzzy Systems* (Vol. 1, pp. 3-6), *2001*.

[Toh et al., 2010] Toh, Kenny Kal Vin, and Nor Ashidi Mat Isa. "Noise adaptive fuzzy switching median filter for salt-and-pepper noise reduction." *IEEE signal processing letters* 17.3 (2010): 281-284.

[Open Source Robotics Foundation, 2014] Open Source Robotics Foundation, "SDF," http://sdformat.org/

[T. Hearn, 2016] T. Hearn, "GitHub - thearn/stl_tools: Python code to produce STL geometry files from plain text, LaTeX code, and 2D numerical arrays (matrices)," https://github.com/thearn/stl_tools

[M. W. Tully Foote, 2017] M. W. Tully Foote, "Robots/TurtleBot - ROS Wiki," http://wiki.ros.org/Robots/TurtleBot