УДК 004; 004.8; 004.9 DOI https://doi.org/10.35546/kntu2078-4481.2023.4.27

І. А. ГОЛОВАТЕНКО
аспірант кафедри інформаційних систем та технологій
Національний технічний університет України
«Київський політехнічний імені Ігоря Сікорського»
ORCID: 0000-0003-0951-5687

А. В. ПИСАРЕНКО
кандидат технічних наук, доцент,
доцент кафедри інформаційних систем та технологій
Національний технічний університет України
«Київський політехнічний імені Ігоря Сікорського»
ORCID: 0000-0001-7947-218X

МЕТОД ПЛАНУВАННЯ МАРШРУТУ В АВТОНОМНИХ ЛОГІСТИЧНИХ КІБЕРФІЗИЧНИХ СИСТЕМАХ ЗАСОБАМИ ШТУЧНОГО ІНТЕЛЕКТУ

У цій статті представлено інноваційний підхід, заснований на алгоритмі A, який містить кілька ключових модифікацій для значного підвищення його функціональності та ефективності в автономній навігації автомобіля. Підхід надає пріоритет безпеці та дотриманню правил дорожнього руху шляхом інтеграції алгоритму A* з механізмом, який контролює безпечну відстань до перешкод. Крім того, введено компонент згладжування траєкторії. Цей компонент покращує кінцевий шлях, створюючи більш плавну та комфортну траєкторію. Виявлення небезпечних ділянок траєкторії є ще одним фундаментальним аспектом запропонованого підходу. Досягається це шляхом застосування кластеризації методом k-середніх, потужного методу машинного навчання. Завдяки кластеризації сегментів траєкторії система може розпізнавати критичні ситуації, такі як різкі повороти та рух смугою зустрічного руху. Виявлення цих сегментів дозволяє вживати проактивних коригувальних дій для перетворення потенційно небезпечних сценаріїв на безпечніші альтернативи. Одним із революційних елементів підходу є впровадження технології навчання з підкріпленням (Reinforcement Learning – RL). Спеціальна модель RL адаптується до динамічних перешкод у режимі реального часу, підвищуючи здатність системи швидко й ефективно реагувати на несподівані ситуації на дорозі. Ця адаптивність є ключовим фактором, що робить автономні логістичні системи більш безпечними та універсальними. Таким чином, метод пропонує комплексне та інтелектуальне рішення для планування маршруту в автономних кіберфізичних логістичних системах. Поєднуючи алгоритм A* із найсучаснішими методами уникнення перешкод, згладжування траєкторії, визначення небезпеки та адаптивності RL, прокладається шлях до безпечнішої, ефективнішої та адаптивнішої автономної логістики. Цей підхід має потенціал для революції в галузі транспортування та доставки, пропонуючи переконливе бачення майбутнього, де автономні транспортні засоби рухатимуться дорогами з найвищим рівнем безпеки, відповідності та ефективності.*

Ключові слова: *кіберфізичні системи, автономні об'єкти, планування маршруту, штучний інтелект, логістика, reinforcement learning.*

I. A. HOLOVATENKO
Postgraduate Student at the Department of Informational Systems
and Technologies
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
ORCID: 0000-0003-0951-5687

A. V. PYSARENKO
Candidate of Technology, Associate Professor,
Associate Professor at the Department of Informational Systems and Technologies
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
ORCID: 0000-0001-7947-218X

METHOD OF ROUTE PLANNING IN AUTONOMOUS LOGISTICS CYBERPHYSICAL SYSTEMS USING ARTIFICIAL INTELLIGENCE

This paper presents an innovative approach based on the A algorithm, which includes several key modifications to significantly improve its functionality and efficiency in autonomous vehicle navigation. The approach prioritizes safety and traffic compliance by integrating the A* algorithm with a mechanism that monitors the safe distance to obstacles. In addition, a trajectory smoothing component is introduced. This component improves the final path, creating a smoother and more comfortable trajectory. Detection of dangerous sections of the trajectory is another fundamental aspect of the proposed approach. This is achieved by applying k-means clustering, a powerful machine learning method. Thanks to the clustering of the trajectory segments, the system can recognize critical situations such as sharp turns and driving in the oncoming traffic lane. Identifying these segments allows proactive corrective actions to be taken to transform potentially dangerous scenarios into safer alternatives. One of the revolutionary elements of the approach is the introduction of Reinforcement Learning (RL) technology. The special RL model adapts to dynamic obstacles in real time, increasing the system's ability to respond quickly and efficiently to unexpected road situations. This adaptability is a key factor that makes autonomous logistics systems more secure and versatile. Thus, the method offers a comprehensive and intelligent solution for route planning in autonomous cyber-physical logistics systems. By combining the A* algorithm with state-of-the-art obstacle avoidance, trajectory smoothing, detection of dangerous segments, and RL adaptability, the path to safer, more efficient, and more adaptive autonomous logistics is being paved. This approach has the potential to revolutionize transportation and delivery, offering a compelling vision of a future where autonomous vehicles will navigate the roads with the highest levels of safety, compliance and efficiency.*

Key words: *cyber-physical systems, autonomous objects, route planning, artificial intelligence, logistics, reinforcement learning.*

Постановка проблеми

Традиційні алгоритми A* відомі своєю ефективністю та точністю у пошуку найкоротшого шляху в різноманітних застосуваннях, від комп'ютерних ігор до робототехніки. Однак у кіберфізичних логістичних системах існує більш тонкий набір критеріїв, яким має задовольняти траєкторія. Просто знайти най-коротший шлях недостатньо. Щоб автономний об'єкт міг безпечно та ефективно пересуватися складною місцевістю, траєкторія має уникати перешкод, дотримуватися смуг розмітки, адаптуватися до динамічного середовища та бути достатньо плавною для забезпечення комфорту та безпеки.

З практичної точки зору проблема автономного керування є дуже актуальною. У таких умовах автономний об'єкт стикається з безліччю проблем, включаючи рухомі перешкоди (інші автономні об'єкти, пішоходи), статичні перешкоди (бар'єри, розділювачі) і складні правила водіння (розмітка смуг руху, заборонені зони). Планування траєкторії також має бути здатним ідентифікувати «небезпечні фрагменти», як-от круті повороти, або випадки, коли транспортний засіб може бути змушений рухатися смугою зустрічного руху, і завчасно адаптувати шлях для зменшення ризиків. Ці складності роблять завдання планування траєкторії в реальних логістичних програмах набагато складнішим, ніж те, що здатні вирішити традиційні алгоритми пошуку шляху.

Удосконалення традиційних алгоритмів A* шляхом включення здатності адаптуватися до складних критеріїв реального світу сприяє більшому науковому об'єму знань у таких галузях, як штучний інтелект, ро-бототехніка та обчислювальна логістика. Цей прогрес може прокласти шлях до більш надійних та інтелектуальних автономних систем, здатних приймати рішення в режимі реального часу, які враховують широкий набір змінних і обмежень. Такі досягнення можуть поширитися на інші сфери, включаючи навігацію безпілотників, складську робототехніку та інші проблеми прийняття рішень у складних середовищах.

З практичної точки зору, модифікований алгоритм A* може революціонізувати роботу автоматизованих систем у системі логістики. Наприклад, у галузі автономних логістичних об'єктів безпечніші та ефективніші алгоритми навігації можуть призвести до значного зменшення кількості аварій на дорогах, споживання палива та часу в дорозі. Подібним чином в управлінні ланцюгом постачання розумніші алгоритми визначення шляху можуть сприяти швидшим і безпечнішим доставкам, тим самим зменшуючи витрати та підвищуючи загальну ефективність.

Прямо вирішуючи ці прогалини та проблеми, дане дослідження не лише додає тонкий рівень складності до відомого алгоритму, але й слугує важливою віхою на шляху до створення більш автономних і безпечніших транспортних систем.

Підводячи підсумок, проблема полягає не просто в пошуку шляху, а в пошуку найбільш «відповідного» шляху з огляду на складний набір обмежень реального світу. Модифікація алгоритму A* для задоволення цих потреб є кроком вперед як у наукових дослідженнях, так і в практичних застосуваннях.

Аналіз досліджень і публікацій

Найперші алгоритми планування шляху, такі як A*, створили платформу планування шляху для автономних об'єктів [1]. Проблема використання цих методів полягає в кінетичних обмеженнях, що не зводяться до геометричних.

Деякі останні роботи вирішують цю проблему, модернізуючи старі методи планування траєкторії та використовуючи кінематику транспортного засобу в процесі планування, такі як RRT та hybrid-A* [1].

Hybrid A* є одним із найефективніших планувальників маршруту для автономних об'єктів. Найперша версія цього планувальника представлена на DARPA Urban Challenge 2007 [1]. Використовуючи цей підхід до планування шляху, найважливішим пунктом, який з'єднує дискретний і безперервний простір станів один з одним, є те, як розширити алгоритм пошуку та вибрати правильні наступні вузли для просування далі. Цей метод реалізує кінематику транспортного засобу для вибору наступних потенційних вузлів.

Тим не менш, вищевказаний метод боровся з деякими проблемами, які робили реалістичне застосування цього методу дещо проблематичним, наприклад, слідування за кількома маршрутними точками та наявність більш реалістичної евристичної функції вартості для A*. Ці проблеми будуть більш помітними на великій карті, де цільова позиція знаходиться дуже близько до початкової позиції, але фактична відстань руху до цільової області досить значна.

Автори у [2] наводять свою модифікацію алгоритму A*, використовуючи криві Рідса-Шеппа. Основною метою використання методу кривих Рідса-Шеппа є підвищення точності та пришвидшення пошуку, оскільки даний метод є добре відомим методом для переходу від конфігурації

$$[x_0, y_0, \theta_0] \tag{1}$$

до іншої конфігурації

$$[x_g, y_g, \theta_g] \tag{2}$$

найкоротшим шляхом. Детальніше про криві Рідса-Шеппа можна знайти в [3]. Як доведено під час моделювання, запропонований алгоритм створював більш плавні маршрути та мав принаймні рівний або нижчий ризик зіткнень з навколишніми перешкодами порівняно з маршрутами, створеними звичайним hybrid-A* методом. Недоліком є те, що час обчислення запропонованого аналітичного розширення збільшується приблизно в десять разів.

У дослідженні [4] описано метод згладжування шляху A* для мобільного роботу. Представлена схема згладила шлях, створений A*, як показано в результатах, але запропонований метод не вирішив проблему часу обчислення шляху.

У [5] представлений метод визначення оптимальної ваги евристичної функції для алгоритму A*, щоб мінімізувати час пошуку шляху алгоритмом за рахунок приведення релевантності батьківського вузла поточного вузла до евристичної функції. Час пошуку шляху зменшується за допомогою оптимальної евристичної функції, але довжина шляху збільшується.

В роботі [6] запропонований вдосконалений алгоритм A* для досягнення плавності шляху включенням коефіцієнта відстані до перешкоди в евристичну функцію для пошуку спільного між довжиною шляху та його безпекою, ігноруючи пошук недійсних вузлів, які знаходяться надто близько до перешкод. Результати показують, що плавність траєкторії покращилася порівняно з традиційним методом.

Гібридний алгоритм запропонований у [7]. Алгоритм спочатку використовує алгоритм Дейкстри для визначення початкового шляху, а потім, у разі потенційного зіткнення з динамічною перешкодою, використовується принцип рухомого вікна для вибору локальної оптимальної цільової точки. Базовий алгоритм A* використовується для пошуку нового шляху від поточного місця розташування до мети. Час повторного планування скорочений порівняно з традиційним, але значного покращення довжини шляху досягти не вдалося.

У [8] представлений гібрид алгоритму A* та методу штучного потенційного поля, для уникання динамічних перешкод і заходження коротшого шляху. Запропонований метод зменшив довжину шляху, але пошук шляху зайняв більше часу.

Дослідження [9] пропонує вдосконалений алгоритм A* і деякі нові методи для подальшого покращення продуктивності. Впроваджена стратегія згладжування траєкторії, для усунення зайвих точок і точок перегину та зменшення частих змін напрямку руху мобільних роботів. Підвищений механізм безпеки, щоб уникати перешкоди. Для скорочення часу пошуку та підвищення ефективності алгоритму додана модель витрат та функція адаптивної вартості.

Розглянуті алгоритми планування шляху, зосереджені на пошуку найкоротшого шляху між початковою та кінцевою точками в заданому середовищі. У цій роботі проводиться розроблення та теоретичне обґрунтування методу, який покращує звичайний алгоритм A* шляхом: уведення обмежень результуючої траєкторії для безпечної навігації; впровадження етапу згладжування траєкторії для її відповідності фізичним особливостям автономного об'єкту; кластеризації методом k-середніх для визначення небезпечних особливостей запланованої траєкторії; та використання моделі навчання з підкріпленням для уникнення перешкод у реальному часі.

Постановка задачі

Задача: перевести автономний рухомий об'єкт з початкового стану

$$s_{start} = (x_{start}, y_{start}), \tag{3}$$

у кінцевий

$$s_{goal} = (x_{goal}, y_{goal}) \tag{4}$$

Обмеження: уникнення перешкод, дотримання смуги руху, врахування фізичних особливостей автономного об'єкту.

Задачу можливо поділити на декілька етапів:

1) первинне планування шляху методом A* з уведенням кількох критеріїв: тримати безпечну дистанцію від перешкод; планувати шлях таким чином, щоб він не перетинав горизонтальну дорожню розмітку; виконати перетворення траєкторії таким чином, щоб вона більше відповідала фізичним особливостям автономного об'єкту;

2) проаналізувати отриманий шлях на наявність небезпечних ділянок, таких як круті повороти або виїзд на смугу зустрічного руху;

3) скоригувати траєкторію відповідним чином, задля уникнення небезпечних ділянок;

4) реагувати на зовнішні зміни на шляху слідування з використанням моделі навчання з підкріпленням (reinforcement learning model).

Етап побудови мапи середовища

Цей розділ присвячений процесу створення комплексної мапи, яку можна використовувати для планування шляху та завдань навігації. Зокрема, використано карти формату OpenDRIVE, прийнятий стандарт для опису дорожніх мереж. Зрештою карта перетворюється у двійковий формат, який розмічає смуги розмітки та межі доріг, що робить її сумісною з алгоритмами планування шляху.

На рис. 1 показана дорожня карта міста у форматі OpenDRIVE.

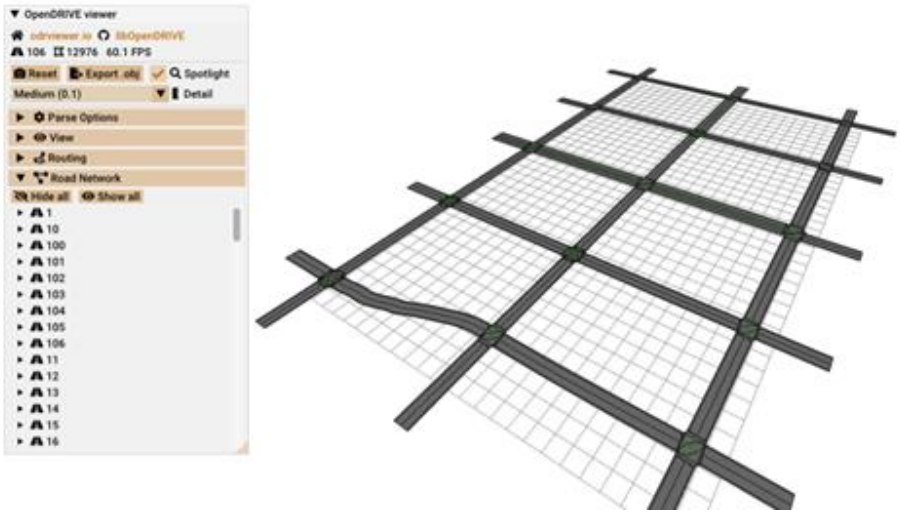

Рис. 1. Приклад мапи місцевості у форматі OpenDRIVE

Етап первинного планування шляху модифікованим A*. Першочергово формується так звана сітка (grid), що являє собою мапу, створену на першому етапі. Далі, отримуючи початкову та кінцеву точку алгоритм починає пошук траєкторії. Нижче наведено лістинги мовою Python модифікованого алгоритму A*.

За основу взятий оригінальний алгоритм A*. Фрагмент коду наведений на рис. 2.

Модифікації зазнали методи, що визначають, чи можна наступну клітинку сітки вносити у результат, чи ні.

Для прикладу, метод is_valid визначає чи не проходить шлях через лінію розмітки зустрічних потоків або чи на достатній відстані від перешкоди знаходиться об'єкт. Клітинка, яка відповідає обом вимога буде додана до результуючої траєкторії. Фрагмент коду наведено на рис. 3.

Результат роботи модифікованого алгоритму наведений на рисунку 4 (представлена частина траєкторії для більшого розуміння результату).

```
def heuristic(self, a, b):
  return np.sqrt((a[0] - b[0])**2 + (a[1] - b[1])**2)

def find_path(self):
  queue = [(0, self.start)]
  came_from = {self.start: None}
  cost_so_far = {self.start: 0}

  while queue:
    _, current = heapq.heappop(queue)

    if current == self.goal:
      break

    directions = [(-1, 0), (1, 0),
                  (0, -1), (0, 1),
                  (-1, -1), (-1, 1),
                  (1, -1), (1, 1)]
    for dx, dy in directions:
      next_cell = (current[0] + dx, current[1] + dy)

      if self.is_valid(next_cell[0], next_cell[1]):
        new_cost = ...

        if next_cell not in cost_so_far \
          or new_cost < cost_so_far[next_cell]:
          ...
          priority = new_cost + self.heuristic(self.goal, next_cell)
          heapq.heappush(queue, (priority, next_cell))
          ...
  ...
```

Рис. 2. Фрагмент коду алгоритму A*

```
def is_valid(self, x, y):
  rows, cols = len(self.grid), len(self.grid[0])
  if 0 <= x < rows and 0 <= y < cols:
    cell_value = self.grid[x][y]

    if cell_value == 1 or self.is_near_obstacle(x, y):
      return False

    if cell_value == 2:
      return False  # Never move onto a lane divider

  return True
return False
```

Рис. 3. Фрагмент коду методу is_valid

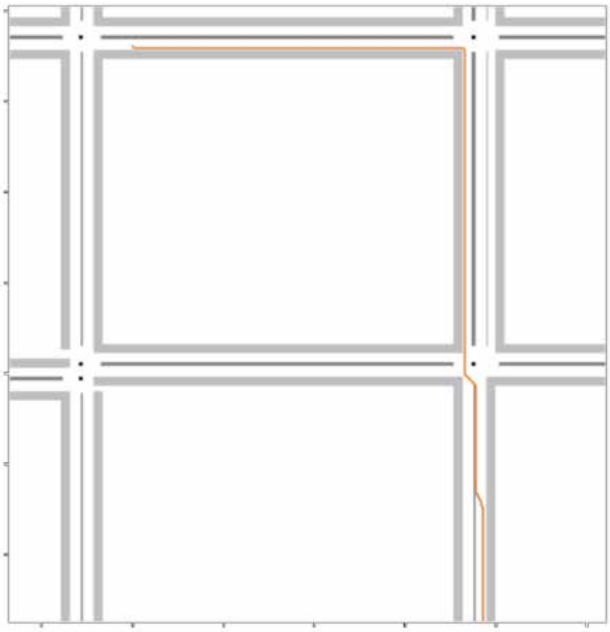

Рис. 4. Фрагмент побудованої траєкторії

Чітко видно, що базовий алгоритм виконує свою задачу, знаходячи шлях у відповідності до евристики.

Також видно, що модифікація алгоритму будує шлях, тримаючись на відстані від перешкод (межі дороги) також не перетинаючи межі розділення зустрічних смуг.

Також очевидно, що наразі траєкторія не бере до уваги логічні особливості смуг руху. Саме тому відбувається виїзд на смугу зустрічного руху.

Етап згладжування первинної траєкторії

Алгоритм бере за основу дані попереднього етапу та модифікує їх у відповідності до задачі.

На рис. 5 наведено фрагмент коду, котрий для кожної точки траєкторії знаходить найближчі точки межі дороги та лінії розділення зустрічних потоків. Далі між отриманими точками визначається середина. І, врешті, відбувається заміна попередньої точки на отриману.

```
def smooth_path(raw_path, grid):
 ...

 for i in range(len(raw_path) - 1):
  ...

  nearby_obstacle = sorted(
   find_nearby_obstacle(grid, x, y, dx, dy)
  )
  new  x, new  y = find  midpoint(
   nearby_obstacle[0], nearby_obstacle[-1]
  )

  smoothed_path.append((new_x, new_y))

 smoothed_path.append(raw_path[-1])

 return smoothed_path
```

Рис. 5. Фрагмент коду алгоритму згладжування траєкторії

Результат роботи модифікованого алгоритму наведений на рис. 6 (представлена частина траєкторії для більшого розуміння результату). Алгоритм чудово впорався із завданням згладивши гострі ділянки поворотів та оновивши траєкторію, тримаючи її по середині смуги руху.

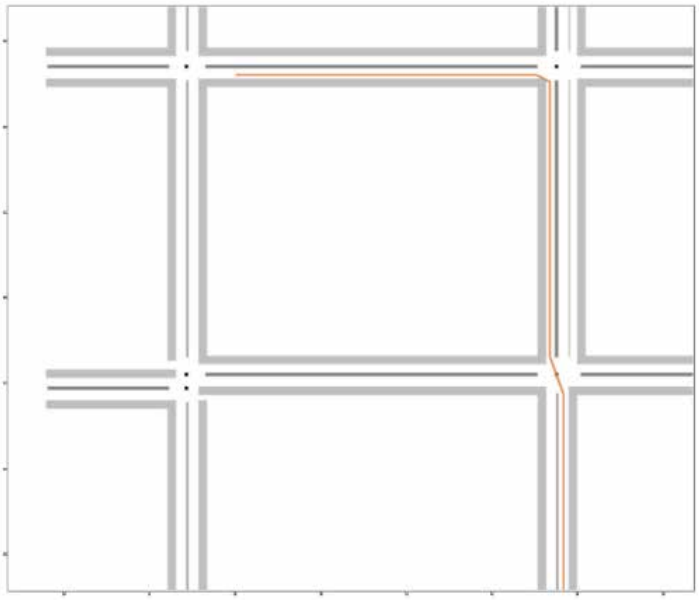

Рис. 6. Фрагмент модифікованої траєкторії

Етап аналізу траєкторії на небезпечні ділянки

Наступний етап перед визначенням остаточної траєкторії для руху автономним об'єктом – визначення небезпечних ділянок у запланованій траєкторії. Ділянки, які потребують найбільшої уваги це, звісно, круті повороти та виїзди на смугу зустрічного руху. Алгоритм виконує так зване видобування «ознак» траєкторії.

На рис. 7 наведено фрагмент коду видобування ознак.

```
def get_path_labels(path, grid):
  ...

  f1 = count_sharp_turns(chunk)
  f2 = detect_oncoming_lane_violations(chunk, grid)
  feature_vector = [f1, f2]
  feature_vectors.append(feature_vector)

  ...

  X = np.array(feature_vectors)
  kmeans = KMeans(n_clusters=3)
  kmeans.fit(X)

  # Get Cluster Labels
  return (chunks, kmeans.labels_)
```

Рис. 7. Фрагмент коду видобування «ознак» траєкторії

В даному випадку видобуваються дві ознаки: гострий кут повороту та виїзду на смугу зустрічного руху.

Визначення гостроти повороту відбувається шляхом знаходження кута між двома векторами. Відповідний фрагмент коду наведено на рис. 8.

```
def angle_between_three_points(A, B, C):
  BA = [A[0]-B[0], A[1]-B[1]]
  BC = [C[0]-B[0], C[1]-B[1]]
  dot_product = BA[0] * BC[0] + BA[1] * BC[1]
  magnitude_BA = math.sqrt(BA[0]**2 + BA[1]**2)
  magnitude_BC = math.sqrt(BC[0]**2 + BC[1]**2)
  magnitude = magnitude_BA * magnitude_BC
  angle = math.acos(dot_product / magnitude)
  return math.degrees(angle)
```

Рис. 8. Фрагмент коду знаходження кута повороту

Перший вектор – траєкторія автономного об'єкту перед поворотом. Другий – траєкторія одразу після закінчення повороту. Шляхом простих алгебраїчних перетворень знаходиться кут між векторами, що порівнюється з граничним значенням. У разу якщо кут менше гранично допустимого значення – поворот вважається гострим.

Друга ознака, що видобувається з траєкторії – виїзд на смугу зустрічного руху. Фрагмент коду наведено на рис. 9.

```
def detect_oncoming_lane_violations(path, grid):
  violations = 0
  for i in range(len(path) - 1):
    ...

    for j in range(1, 6):
      if dx > 0:
        if grid[x1][y1 - j] == 2: violations += 1
      elif dx < 0:
        if grid[x1][y1 + j] == 2: violations += 1
      elif dy > 0:
        if grid[x1 + j][y1] == 2: violations += 1
      elif dy < 0:
        if grid[x1 - j][y1] == 2: violations += 1
    ...

  return violations
```

Рис. 9. Фрагмент коду ідентифікації виїзду на зустрічну смугу руху

Алгоритм шукає смугу розділення зустрічних потоків з правої сторони від запланованої траєкторії, якщо знаходить – сигналізує про порушення, додаючи певну оцінку до сегменту, що обробляється. Зрозуміло, що алгоритм слідує правостороннім правилам дорожнього руху. Однак модифікувати його до потреб ліво-стороннього руху не буде складною задачею.

Результатом алгоритму будуть оцінки, проставлені кожному окремому сегменту шляху, де 0 – буде відповідати безпечному сегменту а все що більше 0 – небезпечному з точки зору правил дорожнього руху та фізичних особливостей автономного об'єкту.

Результат роботи алгоритму наведений на рис. 10 (представлено частину траєкторії для більшого розуміння результату).

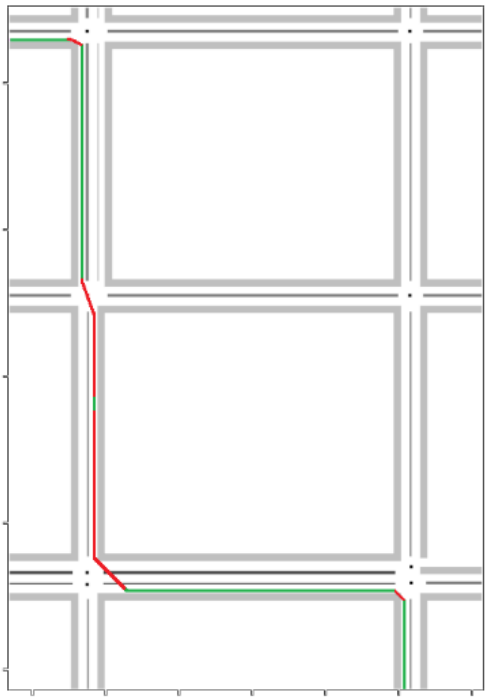

Рис. 10. Фрагмент траєкторії з промаркованими небезпечними ділянками

Алгоритм чудово впорався із завданням віднайшовши частини траєкторії, що вважаються небезпечними з огляду на набір ознак та їх оцінки, що були введені.

Наведені гострі повороти промарковані червоним, що є гарним результатом. Єдина частина траєкторії, що виконувала виїзд на смугу зустрічного руху також промаркована червоним. Однак, невелика ділянка шляху все ж залишилася непромаркованою. Ця проблема має назву «хибно позитивне визначення» (false-positive) або ж «викид» (outlier) алгоритму кластеризації, що не сильно впливає на загальну картину результату.

Етап корекції небезпечних ділянок траєкторії

Визначивши небезпечні ділянки траєкторії, за допомогою методу k-середніх, наступним етапом є застосування коригувальних заходів. На цьому етапі використовуються два методи для вирішення різних типів небезпечних ділянок: базове відображення лінії для усунення проблем із рухом смугою зустрічного транспорту та криві Без'є для згладжування різких поворотів. Ці методи призначені для забезпечення оптимізації кінцевого шляху не лише для ефективності, але й для безпеки та практичності.

Нехай

$$P_{от} = [p_1, p_2, \ldots, p_m], \tag{5}$$

послідовність точок на небезпечній ділянці із зустрічним рухом. Відображення лінії можна математично визначити таким чином:

$$p_i^{'} = p_i - 2 \cdot (p_i n - d) \cdot n, \tag{6}$$

де $p_i^{'}$ – нова точка;

p_i – оригінальна точка;

n – вектор нормалі до горизонтальної розмітки;

d – відстань від центру горизонтальної розмітки по нормалі.

Відображена ділянка

$$P'_{от} = [p'_1, p'_2, \ldots, p'_m], \tag{7}$$

замінює $P'_{от}$ вихідного шляху P.

Проблема різких поворотів вирішується за допомогою кривих Без'є, які згладжують траєкторію.

Кубічна крива Без'є $B(t)$ визначається чотирма контрольними точками

$$B(t) = (1-t)^3 \cdot P_0 + 3 \cdot (1-t)^2 \cdot t \cdot P_1 + 3 \cdot (1-t) \cdot t^2 \cdot P_2 + t^3 \cdot P_3, 0 \le t \le 1, \tag{8}$$

Контрольні точки вибираються на основі точок траєкторії до і після різкого повороту, а також конкретних точок, де відбувається поворот. Крива Без'є замінює оригінальний різкий поворот на шляху.

Після внесення виправлень шлях P повторно оцінюється на наявність будь-яких небезпечних ділянок, що залишилися. Якщо такі знайдені, алгоритм повертається до етапу кластеризації k-середніх, забезпечуючи строгий ітераційний процес для найбезпечнішого та найефективнішого планування шляху.

Результат роботи алгоритму наведений на рис. 11 (представлено частину траєкторії для більшого розуміння результату).

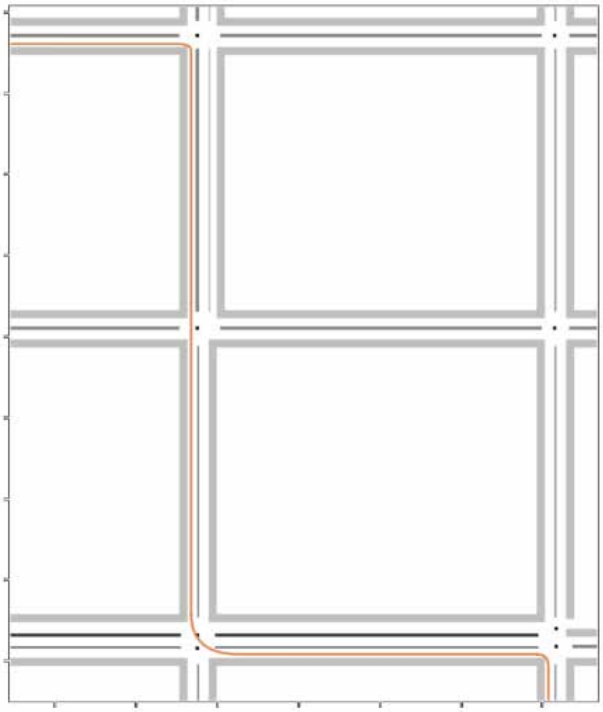

Рис. 11. Фрагмент згладженої траєкторії

Алгоритм чудово впорався із завданням зглядивши гострі кути поворотів та відобразивши частину траєкторії, що проходила смугою зустрічного руху. Траєкторія готова до використання автономним логістичним об'єктом.

Етап адаптивного реагування на завади

У сфері логістики штучний інтелект (ШІ) став ключовим фактором підвищення ефективності, точності та адаптивності операцій у ланцюзі поставок і транспортуванні. Моделі ШІ в логістиці відіграють вирішальну роль у різних аспектах, таких як оптимізація маршрутів, управління запасами, прогнозування попиту та прийняття рішень у реальному часі. Тут глибше досліджуємо значення та застосування моделей AI у логістиці, зосереджуючись на нюансах планування шляху та адаптації в реальному часі:

Алгоритми на основі штучного інтелекту аналізують історичні дані, умови руху та фактори навколишнього середовища, щоб оптимізувати маршрути доставки. Це зменшує споживання палива, підвищує ефективність використання часу та мінімізує експлуатаційні витрати. Зокрема, моделі Reinforcement Learning допомагають адаптуватися до динамічних умов і оптимізувати маршрути в реальному часі.

У той час як попередні етапи гарантують, що шлях є безпечним, ефективним і бере до уваги різні обмеження, динамічна природа реальних середовищ вимагає додаткового рівня адаптивності. Саме тут і потрібна RL модель: вона дозволяє автономному об'єкту коригувати свій шлях у режимі реального часу, щоб уникнути непередбачених перешкод, не вимагаючи повного повторного обчислення всієї траєкторії.

Незважаючи на надійність модифікованого алгоритму A* у створенні початкового безпечного та ефективного шляху, перешкоди можуть з'явитися або змінитися після встановлення шляху. Вони можуть варіюватися від раптово припаркованого автомобіля до пішоходів, які переходять дорогу. Тому механізм, який забезпечує швидку адаптацію в режимі реального часу, має вирішальне значення для практичного застосування.

Модель RL працює в парадигмі «стан-дія-винагорода» [10, 11]. Простір станів складається з поточного положення та орієнтації автономного об'єкту, а також умов навколо нього, таких як перешкоди поблизу. Простір дії включає в себе набір маневрів, які може здійснювати автомобіль, як-от незначні відхилення від траєкторії або

коригування швидкості. Нагороди надаються за успішне уникнення перешкод, зберігаючи загальний шлях, який є максимально близьким до початкової траєкторії.

На рис. 12–16 наведено фрагменти імплементації даного алгоритму з використанням gym [12] як інструменту для опису моделі.

Алгоритм визначає «стан» як положення об'єкту на двовимірній мапі (рис. 12).

```
def step(self, action):
 x, y = self.state

 ...

 return self.state, reward, done, False, {}
```

Рис. 12. Фрагмент коду визначення «стану»

«Дія» визначається набором з восьми варіантів, що представляють собою прості рухи як-от: «ліворуч», «праворуч», «уперед», «назад» та чотири діагональні відповідники (рис. 13).

```
def step(self, action):
 ...

 if action == 0: x = max(0, x-1)
 elif action == 1: x = min(self.grid_size_x-1, x+1)
 elif action == 2: y = max(0, y-1)
 elif action == 3: y = min(self.grid_size_y-1, y+1)
 elif action == 4: x, y = max(0, x-1), max(0, y-1)
 elif action == 5:
  x, y = max(0, x-1), min(self.grid_size_y-1, y+1)
 elif action == 6:
  x, y = min(self.grid_size_x-1, x+1), max(0, y-1)
 elif action == 7:
  x, y = min(self.grid_size_x-1, x+1), \
         min(self.grid_size_y-1, y+1)

 ...

 return self.state, reward, done, False, {}
```

Рис. 13. Фрагмент коду визначення «дії»

«Винагорода» ж в свою чергу визначається у відповідності до виконаної дії за наступним принципом:

1) якщо відбулося досягнення цілі – отримати найвищу винагороду (рис. 14);

```
def _get_reward(self, x, y):
 ...

 if (x, y) == self.end:
  return 50
```

Рис. 14. Фрагмент коду «винагороди» за досягнення цілі

2) відбувся наїзд на смугу розділення чи на перешкоду – отримати високий штраф (рис. 15);

```
def _get_reward(self, x, y):
 ...

 if self.grid[x][y] in [1, 4]:
  return -20
```

Рис. 15. Фрагмент коду «штрафу»

3) маленький штраф, щоб стимулювати систему шукати найкоротше рішення і не стояти на місці (рис. 16).

```
def _get_reward(self, x, y):
 ...
 return -1
```

Рис. 16. Фрагмент коду стимулювання пошуку

На рис. 17 наведений фрагмент траєкторії з перешкодою на шляху прямування автономного об'єкту. Зліва від об'єкту знаходиться межа дороги, а справа – межа розділення зустрічних потоків.

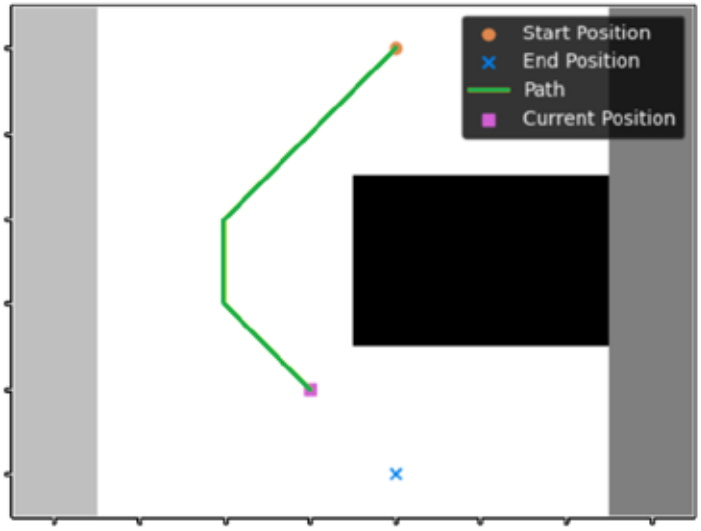

Рис. 17. Приклад об'їзду перешкоди з використанням адаптивного алгоритмом

З рисунку видно, що траєкторія, визначена адаптивним алгоритмом, тримає безпечну дистанцію від перешкоди та меж дороги.

Висновки

Представлена багатоетапна модифікація алгоритму A*, розроблену для розширеного планування траєкторії в умовах логістики за допомогою автономних рухомих транспортних засобів. Метод включає інноваційні елементи, такі як дотримання безпечної дистанції до перешкод, дотримання смуги руху, згладжування траєкторії, ідентифікація та виправлення небезпечних ділянок за допомогою кластеризації k-середніх, а також уникнення перешкод у реальному часі за допомогою навчання з підкріпленням.

Метод демонструє ефективність у створенні траєкторій, які є не тільки оптимальними в традиційному розумінні, але й більше відповідають логістичним обмеженням реального світу. Також метод забезпечує високу адаптивність до непередбачених перешкод в реальному часі.

Незважаючи на те, що модель є багатообіцяючою, все ще є області, які можуть отримати користь від подальших досліджень і оптимізації:

1) обчислювальна ефективність: зі збільшенням складності етапів алгоритму зростає і обчислювальне навантаження. Майбутня робота може бути зосереджена на вдосконаленні алгоритмів, щоб зменшити витрати на обчислення;

2) багатоагентні системи: модель наразі не враховує поведінку інших агентів (наприклад, транспортних засобів) у середовищі. Включення багатоагентної співпраці або конкуренції може зробити модель більш реалістичною;

3) подальше вдосконалення моделі навчання з підкріпленням: компонент можна розширити, щоб враховувати більше характеристик середовища, наприклад сигнали світлофора, для ще більш надійної навігації в реальному часі;

4) більш точна кластеризація: продуктивність k-середніх у ідентифікації небезпечних фрагментів потенційно може бути покращена за допомогою альтернативних методів кластеризації або методів оптимізації параметрів.

Природним наступним кроком є застосування алгоритму до реальних логістичних проблем і автономних навігаційних систем транспортних засобів, що дозволяє оцінювати продуктивність у більш складних і динамічних середовищах.

Нижче наведемо ще декілька перспективних напрямів подальшого дослідження:

1) інтеграція з сенсорними даними: включення сенсорних даних у реальному часі (наприклад, LiDAR, радар та камери) може запропонувати більш повне розуміння навколишнього середовища, зробивши модель ще надійнішою;

2) онлайн-навчання: впровадження механізму онлайн-навчання в модель навчання з підкріпленням може дозволити системі покращувати свою продуктивність з часом, навчаючись на минулому досвіді.

Завдяки подальшому вдосконаленню цих елементів і розширенню моделі для включення додаткових факторів реального світу, вважаємо, що алгоритм має потенціал для значного вдосконалення сучасного рівня автоматизованого планування траєкторії для різних логістичних програм.

Таким чином, ця робота є кроком до створення розумніших, безпечніших і ефективніших алгоритмів планування шляху, які адаптуються до умов реального світу, що постійно змінюються. Майбутні роботи можуть спиратися на цю основу для вирішення ще більш складних і тонких завдань у сфері автономної навігації та логістики.

Список використаної літератури

1. Practical search techniques in path planning for autonomous driving / D. Dolgov та ін. *First International Symposium on Search Techniques in Artificial Intelligence and Robotics*, 2008 р.

2. Improved Analytic Expansions in Hybrid A-Star Path Planning for Non-Holonomic Robots / C. Dang та ін. *Applied Science*. Т. 12, № 12. С. 59–99.

3. LaValle S. M. Planning Algorithms. Cambridge: Cambridge University Press, 2006. 826 с.

4. Path planning and obstacle avoidance approaches for mobile robot / H. Nguyen та ін. *International Journal of Computer Science Issues.* № 3–4.

5. Path Planning of mobile robot based on improved A* Algorithm / M. Lin та ін. *29th Chinese Control and Decision Conference,* 2017 .

6. Improved safety-first a-star algorithm for autonomous vehicles / J. Yu та ін. *5th International Conference on Advanced Robotics and Mechatronics,* 2020 р.

7. Rapid path planning algorithm for mobile robot in dynamic environment / H-m. Zhang та ін. *Advances in Mechanical Engineering.* Т. 9, № 12. С. 1–12.

8. Path Planning using artificial potential field method and A-star fusion algorithm / C. Ju та ін. *Global Reliability and Prognostics and Health Management,* 2020 р, С. 1–7.

9. Improved A* Path Planning Method Based on the Grid Map / Y Ou та ін. *Sensors*. Т. 22, № 16.

10. Barto A. G. Chapter 2 – Reinforcement Learning. Reinforcement Learning: An Introduction. Лондон, 2014. С. 45–67.

11. Ding, Z. та ін. Introduction to reinforcement learning. Deep reinforcement learning: fundamentals, research and applications. 2020. С. 47–123.

12. *OpenAI. Gym library*. URL: https://www.gymlibrary.dev/index.html (дата звернення 10.10.2023).

References

1. D. Dolgov. Practical search techniques in path planning for autonomous driving, 2008 First International Symposium on Search Techniques in Artificial Intelligence and Robotics, 2008.

2. Dang, C. V., Ahn, H., Lee, D. S., & Lee, S. C. (2022, June 13). Improved Analytic Expansions in Hybrid A-Star Path Planning for Non-Holonomic Robots. *Applied Sciences*, *12*(12), 5999. https://doi.org/10.3390/app12125999.

3. LaValle, S. M. (2006, May 29). *Planning Algorithms*. Cambridge University Press.

4. Path planning and Obstacle avoidance approaches for Mobile robot. (2016, July 31). *International Journal of Computer Science Issues*, *13*(4), 1–10. https://doi.org/10.20943/01201604.110

5. M. Lin, K. Yuan, C. Shi and Y. Wang, Path planning of mobile robot based on improved A* algorithm, 2017 29th Chinese Control And Decision Conference (CCDC), Chongqing, China, 2017, pp. 3570-3576, doi: 10.1109/CCDC.2017.7979125.

6. J. Yu, J. Hou and G. Chen, Improved Safety-First A-Star Algorithm for Autonomous Vehicles, 2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM), Shenzhen, China, 2020, pp. 706-710, doi: 10.1109/ICARM49381.2020.9195318.

7. Zhang, H. M., & Li, M. L. (2017, December). Rapid path planning algorithm for mobile robot in dynamic environment. *Advances in Mechanical Engineering*, *9*(12), 168781401774740. https://doi.org/10.1177/1687814017747400

8. C. Ju, Q. Luo and X. Yan, Path Planning Using Artificial Potential Field Method And A-star Fusion Algorithm, 2020 Global Reliability and Prognostics and Health Management (PHM-Shanghai), Shanghai, China, 2020, pp. 1-7, doi: 10.1109/PHM-Shanghai49105.2020.9280929.

9. Ou, Y., Fan, Y., Zhang, X., Lin, Y., & Yang, W. (2022, August 18). Improved A* Path Planning Method Based on the Grid Map. *Sensors*, *22*(16), 6198. https://doi.org/10.3390/s22166198

10. Sutton, R. S., & Barto, A. G. (2018, November 13). *Reinforcement Learning, second edition*. MIT Press.

11. Dong, H., Ding, Z., & Zhang, S. (2020, June 29). *Deep Reinforcement Learning*. Springer Nature.

12. *OpenAI. Gym library*. (2020). Retrieved October 10, 2023, from https://www.gymlibrary.dev/index.html