Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра математики
Факультету інформатики

**Курсова робота**
**за спеціальністю 113 Прикладна математика**
**освітня програма «Прикладна математика»**

# УПРАВЛІННЯ РОБОТОМ-КУР'ЄРОМ В ЗАКРИТИХ ТА ВІДКРИТИХ СИСТЕМАХ

Керівник курсової роботи
асистент Вознюк Я. І.

_____

*(підпис)*

"____" _____ 2022 р.

Виконав студент
3-го року навчання спеціальності
113 "Прикладна математика"
Стефанюк Євген Ігорович
"____" _____ 2022 р

Київ 2022

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри інформатики,
доц., канд.ф.-м.н.
_____С. С. Гороховський

„____”_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студенту 3-го курсу факультету інформатики курсу
Стефанюку Євгену Ігоровичу
ТЕМА Управління роботом-кур'єром в закритих та відкритих системах
Вихідні дані:

- Досліджено проблеми робота в реальному світі та алгоритм D* Lite для динамічних середовищ.

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Календарний план
Анотація
Вступ
1 Проблеми, з якими може зіткнутися робот, та способи їхнього вирішення
2 Виявлення ям на дорозі
3 Проблема з дверми
4 Сходи й бордюри
5. Нерухомі та рухомі об’єкти та способи їхнього виявлення
6. Інші проблеми
7. Алгоритм для пошуку шляху в частково невідомому графі
Висновки
Список літератури

Дата видачі „___” _____ 2022 р. Керівник _____
(підпис)

Завдання отримав_____
(підпис)

**Тема:** Управління роботом-кур'єром в закритих та відкритих системах

**Календарний план виконання роботи:**

| Номер | Назва етапу курсової | Термін виконання етапу | Примітка |
|---|---|---|---|
| 1. | Отримання завдання на курсову роботу. | жовтень | |
| 2. | Огляд технічної літератури за темою роботи. | листопад | |
| 3. | Аналіз методів для виявлення ям. | грудень-січень | |
| 4. | Аналіз проблем з дверми та сходами. | лютий | |
| 5. | Аналіз способів виявлення рухомих та нерухомих перешкод. | березень | |
| 6. | Аналіз інших проблем. | кінець березня | |
| 7. | Аналіз наявних алгоритмів для динамічних середовищ. | квітень | |
| 8. | Текстове оформлення результатів роботи. | травень | |
| 9. | Попередній аналіз курсової. Виправлення помилок. | кінець травня | |
| 10. | Захист курсової роботи. | червень | |

Студент Стефанюк Є. І.

Керівник Вознюк Я. І.

“_____” _____

*Зміст*

# Анотація

Метою цієї курсової було дослідити проблеми, з якими роботи-кур’єри можуть зіткнутися в реальному житті. Проаналізовано метод для виявлення ям на дорозі. Окремо розглядається алгоритм для пошуку шляху в невідомому графі для динамічних систем.

**Ключові слова:** робот-кур’єр, алгоритм, метод.

# Вступ

Світ прямує до автоматизації ще з часів промислової революції. Впровадження у виробництво механізмів дозволило значно збільшити потужності заводів та зменшити ручну працю людей. У двадцять першому сторіччі тенденція на автоматизацію продовжилась і активно триває. Всюди, де це можливо, людей замінюють роботи. Тепер машина може робити операції, обслуговувати людей, спілкуватись, керувати машиною чи навіть літаком.

Однак на сьогодні ще багато галузей недостатньо чи взагалі не автоматизовані, хоча теоретично роботи там можуть бути. Кур’єри – теж не виключення. Крім використання доставки в закритих системах, як склади, де постійно треба переміщувати якісь об’єкти, зараз активно розвиваються служби доставки. Часто службам не вистачає потужності та доставка може тривати довгий час, тому створення робота-кур’єра допоможе компенсувати відсутність вільних людей та пришвидшити кур’єрські послуги. Зараз активно йде робота над роботами-кур’єрами, деякі компанії їх активно тестують, але на сьогодні все ще немає широкого застосування таких роботів – у наявних екземплярів присутні недоліки, які не дають їх використовувати.

Виходячи з цих тенденцій, за мету даної роботи були поставлені задачі дослідити наявні методи, які можуть бути застосовані для роботи робота-кур’єра, модифікувати їх та запропонувати свої розв'язання проблем, які можуть виникнути під час розробки кур’єра.

Використання запропонованих рішень дозволить скоротити час на розробку та налаштування роботів. Це наблизить нас до широкого застосування та автоматизацію в цій сфері.

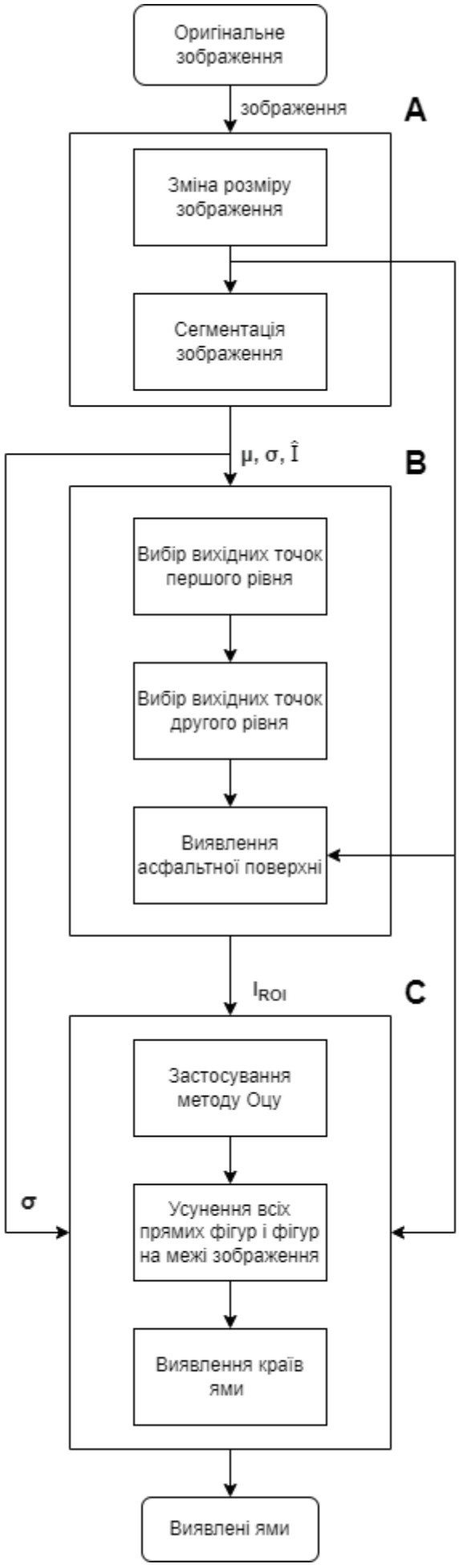
# 1 Проблеми, з якими може зіткнутися робот

Хоча й відбувається успішне тестування і навіть подекуди використання роботів-кур’єрів, масового їх застосування в реальному світі поки що нема. Проблема в тому, що хоч в нас і є алгоритми пошуку шляху і планування руху для роботів в теорії, застосування роботів на практиці – інша річ. З закритими системами простіше – набагато менше чинників, які треба брати до уваги, однак теж є над чим попрацювати – сходи, нерівності поверхні, двері, здебільшого різноманітні нерухомі перешкоди, такі як коробки, стелажі та тому подібне. У відкритих же системах кількість різних факторів зростає в рази. Роботи-кур’єри їздитимуть пішохідними переходами, тому їм потрібно оминати окрім нерухомих, ще й рухомі об'єкти - людей, тварин тощо, отже наш робот має вміти оцінювати динамічну ситуацію навколо нього і реагувати на неї.  Окрім вищенаведених, потрібно вміти розбиратися з закритими дверми та хвіртками, домофонами в будинках, ліфтами тощо. Також важливий чинник – погода. Робот повинен вміти орієнтуватися в дощову погоду, коли падає сніг, і в темряві.

# 2 Виявлення ям на дорозі

Передусім робот повинен вміти визначати, можна проїхати певною ділянкою дороги чи ні. Якщо на шляху виникла яма, то робот мусить її повністю об’їхати. Для визначення стану дороги в основному використовують 3 методи - 3D реконструювання, датчики вібрації й аналіз за допомогою камер[10]. 3D реконструювання відбувається за допомогою лазерних сканерів високої вартості[2] і не підходить нам, оскільки для масового використання роботів-кур’єрів ціна має бути прийнятною. Цю проблему можна вирішити за допомогою акселерометрів[3][5], які визначатимуть відстань до поверхні, і на основі цих даних робот розумітиме, безпечно проїжджати дорогою чи ні, а також камер[4], направлених донизу, зібрані зображення з яких можна аналізувати, наприклад, за допомогою детектора границь Кенні[1].

Використаємо спектральне кластерування й обробку зображень[6] для оцінки дороги. Схематично це виглядає так:


Оригінальне зображення
зображення
A
Зміна розміру зображення
Сегментація зображення
μ, σ, Î
B
Вибір вихідних точок першого рівня
Вибір вихідних точок другого рівня
Виявлення асфальтної поверхні
$I_{ROI}$
C
Застосування методу Оцу
σ
Усунення всіх прямих фігур і фігур на межі зображення
Виявлення країв ями
Виявлені ями

A – зображення перед обробкою, B – витягнення області інтересу, C – виявлення ями

Рис. 1 – Діаграма методу для виявлення ям

Перший крок – змінити зображення. Розмір зображення не впливає на можливість виявлення ям, тому для економії ресурсів, необхідних для цього методу, зменшуємо розмір зображення. Далі сегментуємо наше зображення. Для цього знаходимо середнє значення μ (формула 1.1) і середнє квадратичне відхилення σ (формула 1.2).

$$\mu = \frac{1}{3 \cdot \mathrm{N} \cdot \mathrm{M}} \cdot \sum_x \sum_y \sum_{[R,G,B]} p(x,y)_{[R,G,B]} \tag{1.1}$$

$$\sigma = \sqrt{\frac{1}{3 \cdot \mathrm{N} \cdot \mathrm{M} - 1} \cdot \sum_x \sum_y \sum_{[R,G,B]} [p(x,y)_{[R,G,B]} - \mu]^2}\,, \tag{1.2}$$

де N – висота зображення;

M – ширина зображення;

x – індекс рядка;

y – індекс стовпчика;

[R, G, B] – компоненти кольору RGB (червоний, зелений, синій) зображення I.

$p(x,y)_{[R,G,B]}$ відображає піксельні значення для точки (x, y). Першим кроком всім пікселям зображення $I$, які задовольняють рівнянням 1.3, 1.4, 1.5, присвоюємо значення RGB (0, 0, 0) (білий колір). Якщо ж вони не задовольняють їм, то піксель залишаємо незмінним:

$$I_{G(x,y)} - \frac{I_{R(x,y)} + I_{B(x,y)}}{2} + 15 > 0 \tag{1.3}$$

$$I_{R(x,y)} - I_{B(x,y)} > 20 \quad (1.4)$$

$$I_{G(x,y)} - I_{R(x,y)} > 15 \quad (1.5)$$

Наступним кроком знову всім пікселям з зображення $I'$, які задовольняють рівнянням 1.6, 1.7, присвоюємо значення RGB (0, 0, 0):

$$I'_{[R,G,B](x,y)} > (255 - \sigma) \quad (1.6)$$

$$I'_{[R,G,B](x,y)} < \sigma \quad (1.7)$$

У фінальному кроці присвоюємо пікселю зображення $I''$ значення RGB (255, 255, 255) (чорний колір), якщо він задовольняє рівнянню 1.8, чи (0, 0, 0), якщо він задовольняє рівнянню 1.9:

$$I''_{[R,G,B](x,y)} > 0 \quad (1.8)$$

$$I''_{[R,G,B](x,y)} = 0 \quad (1.9)$$

В результаті маємо отримати зображення $\hat{I}$, яке зображено на рисунку 2 (б).

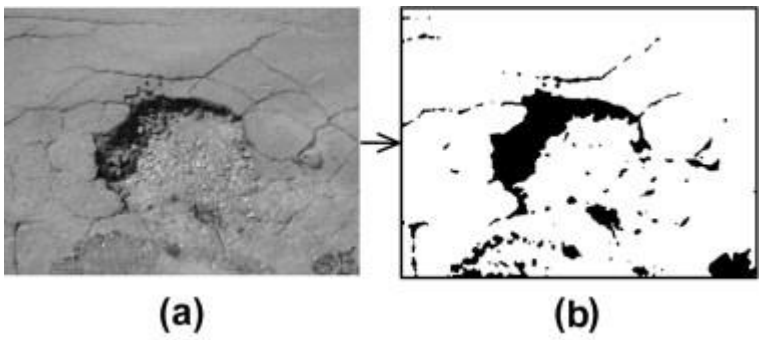



Рис. 2 – Зображення ділянки з ямою до обробки і після.

Після цього переходимо до витягнення області інтересу. Область інтересу в нашому випадку – ділянка дороги, де є яма. Спершу формуємо масив вихідних точок першого рівня. Для цього ми випадковим чином вибираємо кількість пікселів, яка дорівнює $\frac{\sigma}{2}$, з усіх білих пікселів зображення $\hat{I}$. Для точнішого визначення області інтересу формуємо масив вихідних точок другого рівня. Для цього беремо 4 квадрати, сусідні з вихідною точкою першого рівня, зі стороною a, яка задається за формулою 1.10:

$$a = \begin{cases} \frac{\sigma}{5}, \sigma > 60 \\ \frac{\sigma}{3}, \sigma \leq 60 \end{cases} \quad (1.10)$$

Вихідні точки другого рівня обчислюються шляхом порівняння середнього значення кожного квадрату з 250. Якщо це значення більше або рівне за 250, то вихідною точкою другого рівня стає вершина квадрата, яка розташована якнайдалі від нашої вихідної точки першого рівня. Повторюємо цей крок n разів для охоплення якомога більшої площі. На основі цих точок визначаємо нашу область інтересу $I_{ROI}$.

Зрештою переходимо до останнього етапу – виявлення ями. Передовсім ми використаємо метод Оцу. Основна ідея методу – розділити гістограму зображення на два кластери – пікселів переднього та заднього плану – з порогом, що визначається як результат мінімізації зваженої дисперсії цих класів, що позначається як $\sigma_\omega^2(t)$[7]. Цей поріг шукається за формулою 1.11:

$$\sigma_\omega^2(t) = \omega_0(t) \cdot \sigma_0^2(t) + \omega_1(t) \cdot \sigma_1^2(t), \quad (1.11)$$

де $\omega_0, \omega_1$ – ймовірності двох класів, розділених порогом $t$;

$\sigma_0^2, \sigma_1^2$ – відхилення цих двох класів.

Рис. 3 – Приклад зображення з використанням методу Оцу.

Клас ймовірностей $\omega_{0,1}(t)$ обчислюється з ймовірністю $p$ для кожного значення пікселя в двох окремих кластерів $C_0, C_1$ з $L$ гістограм:

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i) \tag{1.12}$$

$$\omega_1(t) = \sum_{i=t}^{L-1} p(i) \tag{1.13}$$

Далі для двох класів показуємо, що мінімізація дисперсії всередині класу рівна максимізації міжкласової дисперсії[7]:

$$\sigma_b^2(t) = \sigma^2 - \sigma_\omega^2(t) = \omega_0(t)\cdot(\mu_0 - \mu_T)^2 + \omega_1(t)\cdot(\mu_1 - \mu_T)^2 =$$

$$= \omega_0(t)\cdot\omega_1(t)\cdot(\mu_0(t) - \mu_1(t))^2, \tag{1.14}$$

яка виражається в термінах $\omega_i$ і середнього значення $\mu_i$, які самі по собі можуть оновлюватися ітеративно. В нашому випадку $\mu_0(t)$, $\mu_1(t)$ і $\mu_T$ дорівнюють:

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} i\cdot p(i)}{\omega_0(t)} \tag{1.15}$$

$$\mu_1(t) = \frac{\sum_{i=t}^{L-1} i\cdot p(i)}{\omega_1(t)} \tag{1.16}$$

$$\mu_T = \sum_{i=0}^{L-1} i\cdot p(i) \tag{1.17}$$

За допомогою цього в нас виникає ефективний алгоритм.

Наступний крок – усунути всі прямі фігури і фігури на межі зображення. Знаходимо фігури за їхнім ексцентриситетом $\epsilon$ і усуваємо за допомогою функції regionprops[8].

Зрештою в нас залишається лише область інтересу, де є яма. На основі виявлених нами фігур, малюємо навколо них прямокутники і визначаємо нашу яму.

# 3 Проблеми з дверми

Перейдемо до закритих дверей і хвірток. Поки що не існує гарних і відносно дешевих методів вирішення цієї проблеми. Одним з рішень може бути встановлення спеціальної роботизованої руки на робота для відкривання дверей різного типу. Однак ще не винайдено такого алгоритму, який би відкривав усе, тому що існує безліч різних замків, дверних ручок, засувів і так далі. Та зрештою, найбільшою проблемою залишається ціна – встановлення такої роботизованої руки збільшить вартість виробництва і програмування одного робота в декілька разів, чого ми хочемо уникнути. Тому поки що наш робот-кур’єр розглядає ці конструкції як перешкоди і прокладає шлях так, аби їх об’їхати. Він розраховуватиме цей маршрут на основі супутникових зображень, тому в нього має бути GPS-навігація. Однак люди живуть в квартирах і домах, які зазвичай мають двері чи/і хвіртки. Що робити в такому випадку? Найкраще й найвигідніше, що ми можемо зараз придумати – робот зупиняється біля будинку нашого клієнта і посилає сигнал про прибуття. Тоді наш клієнт має вийти й забрати свою посилку. Це не всім підходить, наприклад, людям з інвалідністю, однак це компроміс між вартістю і зручністю.

# 4 Сходи й бордюри

У роботів завжди виникали проблеми зі сходами, однак зараз зроблено доволі багато роботів, які можуть ними підніматись[9]. Є багато різних типів роботів – двоногі, чотириногі, гібриди з колісними ногами, гусеничні, які вміють підніматися сходами.

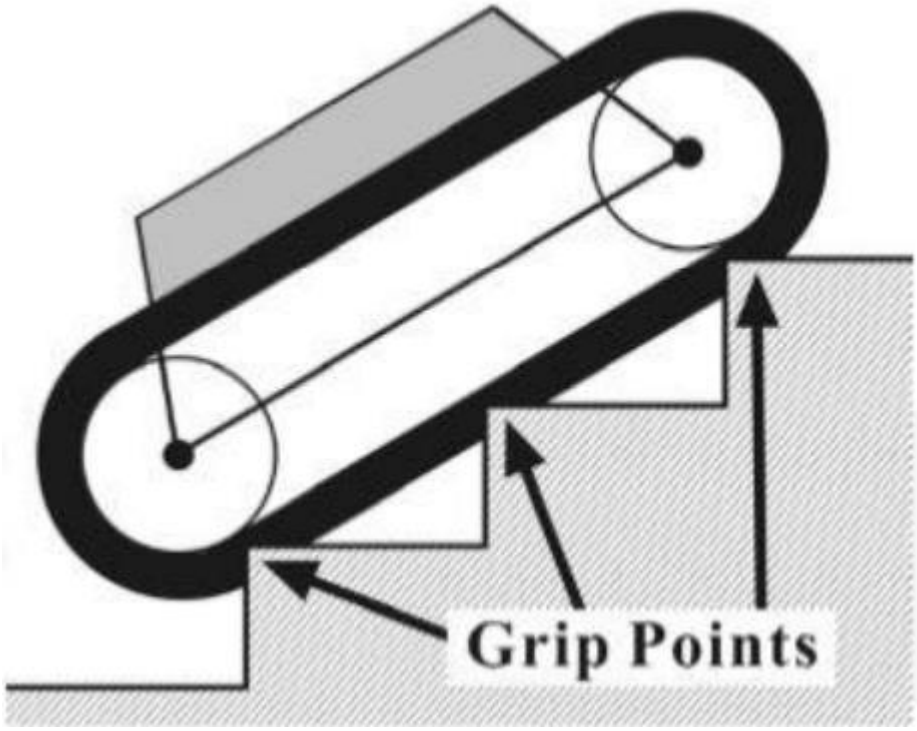



Рис. 4 – Приклад гусеничного робота, що піднімається сходами.

Однак такі роботи не підходять нам, тому що вони недостатньо швидкі – оскільки в нас робот доставляє різні продукти, то клієнт хоче одержати посилку якнайшвидше. Також деякі з цих роботів недостатньо стійкі (наприклад, двоногі) для доставлення пакунків. Тому робимо нашого робота на шасі – чотирьохколісному чи шестиколісному. Для того, щоб наш робот

міг підніматися сходами, можна зробити так, щоб шасі у робота було незафіксованим. Таким чином можна заїжджати і на високі бордюри.


Рис. 5 – Приклад робота-кур’єра з незафіксованим шасі.

# 5 Нерухомі і рухомі об’єкти та способи їхнього виявлення

Тепер треба подумати, за допомогою чого саме ми виявлятимемо перешкоди перед собою. Ми це можемо робити за допомогою ультразвукових датчиків LIDAR (розшифровується як «виявлення та визначення дальності за допомогою світла»). Фактично він працює трохи як радар і гідролокатор[11].

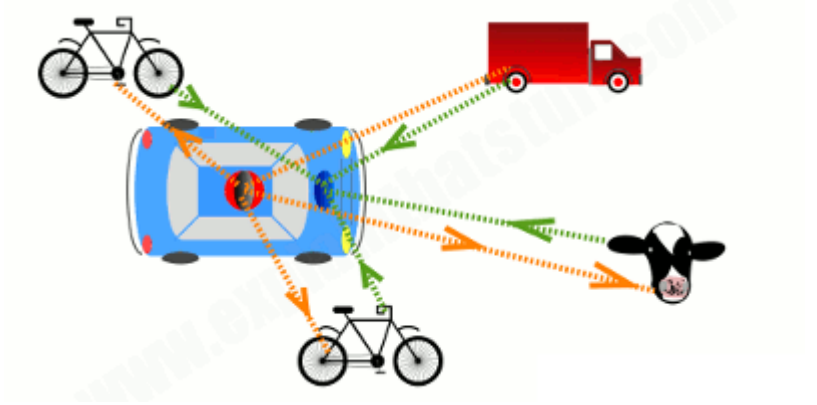


Рис. 6 – Основна ідея LIDAR – автомобіль бачить, посилаючи обертовий лазерний промінь (помаранчева лінія) до перешкод та виявляючи їхні відображення (зелений промінь).

Датчик LIDAR складається з лазера і фотодетектора, який вловлює його відбите світло. Ці датчики допомагають нам орієнтуватися як навколо нерухомих об’єктів (стовпи, стелажі, стіни), так і навколо рухомих (пішоходи тощо) відбиваючи обертовий лазерний промінь від перешкод. Вони використовують час, за який промінь повертається, щоби визначити, наскільки віддалений об’єкт. Таким чином створюється тривимірна картинка динамічного середовища, на основі якої ми можемо змінювати маршрут робота відповідно до наших потреб. Також треба GPS-приймач, щоб оцінити наше місцерозташування. Використовуючи цю технологію, робот, наприклад, сканує обидві сторони перед переходом вулиці, щоб уникнути

зіткнення з транспортними засобами. Також варто згадати, що датчики LIDAR прості, надійні і відносно недорогі, тому чудово нам підходять.

Також за допомогою цих датчиків робот може орієнтуватись в темряві та в дощову погоду, хоча їхня ефективність в погану погоду не надто висока[12][13][14].

# 6 Інші проблеми

Ну й наостанок – проблеми, які виникають більшою мірою через людей, а не через робота. По-перше, робота можна просто викрасти – він їде достатньо повільно для того, щоби не вдарятися в пішоходів, тому його достатньо легко підібрати. Цієї проблеми можна позбутися, вмонтувавши в нього сирену, яка гучно гратиме і привертатиме увагу до крадія, якщо той підніме робота. Також місцезнаходження робота можна відстежити через GPS. По-друге, все ще існують проблеми з тим, що роботи займають доволі багато місця, через що на вузькій вуличці розминутися з пішоходом буде дуже важко. Наприклад, в 2019 стався інцидент, коли один з таких роботів-кур’єрів заблокував перехід дороги людині в інвалідному візочку[15]. Це доводить, що ще є проблеми, пов’язані з безпекою, які треба вирішити.

# 7 Алгоритм для пошуку шляху в частково невідомому графі

Однак немає значення, чи зможе наш робот виявити нерухомі чи рухомі перешкоди, якщо не буде алгоритму, за допомогою якого він проаналізує цю інформацію і прокладе шлях, щоб їх об'їхати. Тому використаємо алгоритм пошуку D* Lite, який можна використати для частково відомого або зовсім невідомого навколишнього середовища. Цей алгоритм заснований на LPA*, який використовує евристичну функцію для перевірки вузлів, які ймовірніше приведуть до цілі, але відрізняється тим, що є динамічним – ціна вузлів може змінюватись під час виконання алгоритму, себто робот може змінювати маршрут, виявивши нову перешкоду. Також ми зазначаємо, що наш робот може рухатися в вісьмох напрямках (Рис. 1.7).

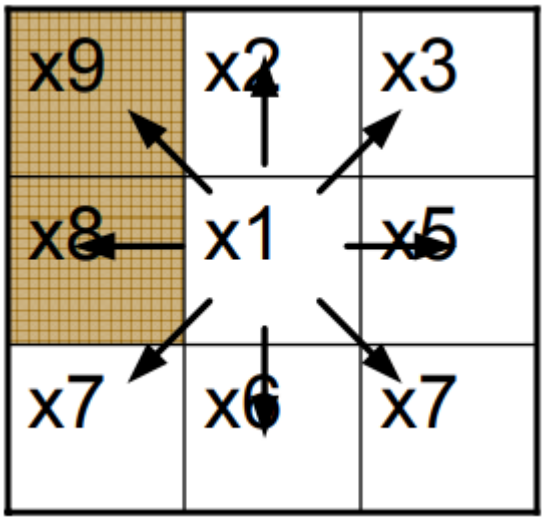



Рис. 7 – Клітинна решітка для планування шляху.

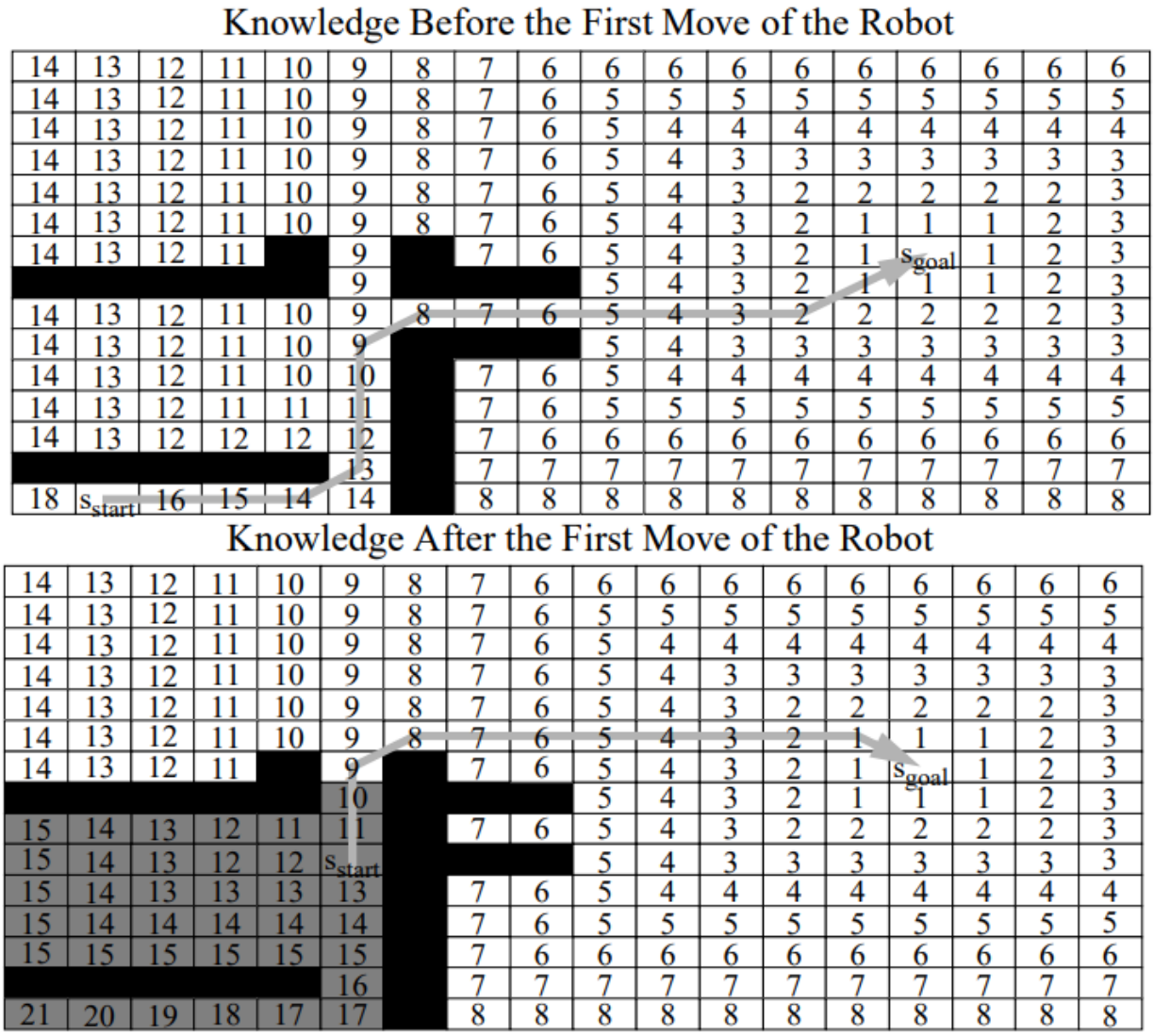



Рис. 8 – Приклад алгоритму D* Lite.

D* Lite, на відміну від LPA*, який шукає найкоротший шлях від початкового вузла до кінцевого, починає від кінцевого вузла до початкового[16]. Для кожного вузла існує функція $h(s, s_{\text{кінцеве}})$, яка визначає ціну шляху від цього вузла до кінцевого. Ціна шляху $h(s_{\text{кінцеве}}, s_{\text{кінцеве}})$ для кінцевого вузла рівна 0. Також в алгоритму є дві оцінки початкової відстані $g^*(s)$ для кожного вузла. Функція відстані шляху до вершини $g(s)$ в алгоритмі D* Lite є вартістю переходу від кінцевого вузла до поточного. Також кожен вузол (за винятком початкового та кінцевого) має n наступників:

- Будь-який вузол, з якого якесь ребро веде до $n$, є попередником $n$:

$$Pred(s) \subseteq S,\ s \in S \tag{7.1}$$

- Будь-який вузол, до якого якесь ребро веде від $n$, є наступником $n$:

$$Pred(s) \subseteq S,\ s \in Sa \tag{7.2}$$

Друга оцінка – $rhs(s)$ – попереднє значення, засноване на значеннях $g$ і потенційно точніше. Знаходимо його за формулою 7.3.

$$rhs(s) = \begin{cases} 0,\ \text{якщо } s = s_{\text{початкове}} \\ min_{s' \in Pred(s)}(g(s') + c(s', s),\ \text{якщо } s \neq s_{\text{початкове}} \end{cases} \tag{7.3}$$

Вузол називають локально узгодженим, якщо його $g(s) = rhs(s)$. Однак після зміни ціни якогось попередника, вузол стає локально неузгодженим і потребує переоцінки. Для цього, як і в алгоритмі LPA*, в D* Lite є пріоритетна черга, в якій зберігають локально неузгоджені вузли. Ця черга відсортована за двовимірним ключем $k(s)$ – спочатку за $k_1$, а потім за $k_2$ в лексикографічному порядку :

$$k(s) = \begin{bmatrix} k_1(s) \\ k_2(s) \end{bmatrix} = \begin{bmatrix} min\,(g(s), rhs(s) + h(s, s_{\text{кінцеве}}) \\ min\,(g(s), rhs(s) \end{bmatrix} \tag{7.4}$$

Розраховуємо найкоротший шлях таким чином:

- починаємо з кінцевого вузла $s_{\text{кінцеве}}$.

- переходимо до попередника $s$, в якого найменший $g(s') + c(s', s)$. Якщо робот виявляє зміни ціни вузлів, від першого елемента пріоритетів кожного вузла черги пріоритетів треба відняти $h(s, s')$. Після цього кожного разу, коли змінюється ціна вузлів, до перших компонентів вузлів треба додавати $h(s, s')$. Якщо наш робот рухається і знову виявляє, що ціна вузлів змінилася, константи сумуються, і ми повторюємо цей крок знову. Після видалення вузла з найменшим пріоритетом, ми переоцінюємо його пріоритет.
- Повторюємо попередній крок, доки не досягнемо цілі.

# Висновки

У цій роботі було досліджено актуальну на сьогодні тему, пов’язану з робототехнікою. Роботи-кур’єри активно розвиваються, тому було прийнято рішення бути частиною цього розвитку та запропонувати рішення, які можуть покращити роботів і зробити їх більш простими.

У процесі виконання роботи було досліджено різні методи, якими зараз користуються в робототехніці. У багатьох напрямах немає стандартних рішень, які б працювали в більшості умов, тому на кожен випадок потрібно робити щось нове або сильно модифікувати наявні рішення. Всі рішення проблем робились максимально простими і зрозумілими, щоб їх можна було адаптувати та модифікувати під різні проблеми.

Попри те, що рішення, які були описані в роботі, добре працюють та мають багато переваг в нашій ситуації, тема не може вважатись закритою. Простір для досліджень залишається великим, стандартних рішень багатьох проблем немає, тому потрібно над цим багато працювати, а основою для цього може слугувати ця робота.

# Список використаної літератури


1. Canny, J. A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. 1986, PAMI-8, 679–698.
2. Kim, T.; Ryu, S.K. Review and analysis of pothole detection methods. J. Emerg. Trends Comput. Inf. Sci. 2014, 5, 603–608.
3. A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, L. Selavo, Real time pothole detection using Android smartphones with accelerometers, Distributed Computing in Sensor Systems and Workshops (DCOSS), pp. 1-6, 2011
4. Staniek, Marcin. ”Stereo vision techniques in the road pavement evaluation.” XXVIII International Baltic Road Conference. 2013.
5. J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, H. Balakrishnan, The pothole patrol: using a mobile sensor network for road surface monitoring, Proceeding of the 6th international conference on Mobile systems, applications, and services, pp. 29-39, 2008
6. Buza, Emir, Samir Omanovic, and Alvin Huseinovic. ”Pothole detection with image processing and spectral clustering.” Proceedings of the 2nd International Conference on Information Technology and Computer Networks. 2013.
7. N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, Jan. 1979.
8. https://uk.mathworks.com/help/images/ref/regionprops.html#bqkf8id.
9. https://singularityhub.com/2009/05/05/robots-take-to-the-stairs-this-is-just-the-beginning-videos/.
10. Buza, E.; Omanovic, S.; Huseinovic, A. Pothole detection with image processing and spectral clustering. In Proceedings of the 2nd International Conference on Information Technology and Computer Networks, Athens, Greece, 27–28 April 2013; pp. 48–53.

11. https://www.explainthatstuff.com/lidar.html
12. M. Kutila, P. Pyykönen, H. Holzhüter, M. Colomb and P. Duthon, "Automotive LiDAR performance verification in fog and rain," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 1695-1701.
13. A. Filgueira, H. González-Jorge, S. Lagüela, L. Díaz-Vilariño, P. Arias, Quantifying the influence of rain in LiDAR performance, Measurement, vol. 95, 2017, pp. 143-148.
14. Abdo, J., Hamblin, S., and Chen, G., "Effective Range Assessment of Lidar Imaging Systems for Autonomous Vehicles Under Adverse Weather Conditions With Stationary Vehicles." ASME. *ASME J. Risk Uncertainty Part B.* September 2022; 8(3): 031103.
15. https://www.bloomberg.com/news/articles/2019-11-19/why-tech-needs-more-designers-with-disabilities
16. Koenig S, Likhachev M. Fast replanning for navigation in unknown terrain. IEEE Transactions on Robotics. 2005.