МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Т.в.о.завідувача кафедри КНІ
_____ Сафонова С.О.
«____»_____ 2020 р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

ТЕХНОЛОГІЇ АНАЛІЗУ І ВІДНОВЛЕННЯ ТРИВМІРНОЇ ГРАФІЧНОЇ ІНФОРМАЦІЇ НА ОСНОВІ ДВОВИМІРНИХ ЗОБРАЖНЬ

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність: 123 - Комп'ютерна інженерія

Науковий керівник роботи:	_____ (підпис)	І.С. Скарга-Бандурова (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Я.О. Критська (ініціали, прізвище)
Студент:	_____ (підпис)	О.О. Шаповалов (ініціали, прізвище)
Група:		КІ-18ДМ

Сєвєродонецьк 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень Магістр
Напрям підготовки 123 - Комп'ютерна інженерія
(шифр і назва)
Спеціальність 123 - Комп'ютерна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:

Т.в.о. завідувача кафедри КНІ
_____ С.О. Сафонова
«_____» _____2020 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Шаповалов Олександр Олександровчи
(прізвище, ім'я, по батькові)

1. Тема роботи Технології аналізу і відновлення тривимірної графічної інформації на основі двовимірних зображень

керівник проекту (роботи) Скарга-Бандурова Інна Сергіївна, д.т.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " " 2019 р. № ____
2. Строк подання студентом роботи 23.01.2020
3. Вихідні дані до роботи Матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз сучасних підходів до тривимірної реконструкції, методів формування об'ємних зображень; Дослідження алгоритмів обробки, визначення ключових характеристик методів: обчислювальна складність, точність побудови моделі; Оцінка ефективності методів; Розробка алгоритму тривимірної реконструкцій із хмари точок на основі методу стерео зіставлення; Тестування алгоритму.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4 Охорона праці та безпека в надзвичайних ситуаціях	Критська Я.О. ст. викл.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Проведення аналізу напрямків які займаються обробкою зображення	02.09.2019 - 15.09.2019	
2	Аналіз основних принципів тривимірної реконструкції	16.09.2019 – 30.09.2019	
3	Розробка розділу «Охорона праці та безпека в надзвичайних ситуаціях»	01.10.2019 – 05.10.2019	
4	Дослідження відомих методів і алгоритмів побудови об'ємної моделі	06.10.2019 – 20.10.2019	
5	Розробка алгоритму тривимірної реконструкцій із хмари точек на основі методу стерео зіставлення	20.10.2019 – 10.11.2019	
6	Програмна реалізація алгоритму на мові С ++ з використанням бібліотеки OpenGL	11.11.2019 – 10.12.2019	
7	Проведення тестування алгоритму	11.12.2019 – 20.12.2019	
8	Оформлення пояснювальної записки	20.12.2019 – 22.01.2020	

Студент _____ Шаповалов О.О.
(підпис) (прізвище та ініціали)

Науковий керівник _____ І.С. Скарга-Бандурова
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Шаповалов А.А. Технології аналізу і відновлення тривимірної графічної інформації на основі двовимірних зображень.

Розглянуто методи та системи обробки зображення, основні принципи тривимірної реконструкції. Проведено аналіз сучасних методів і алгоритмів тривимірної реконструкції. Розроблено алгоритм на основі методу стерео зіставлення і реконструкцій об'єкта з хмари точок. Алгоритм дозволяє реконструювати тривимірний об'єкт з набору двовимірних зображень з високою точністю.

Ключові слова: тривимірна реконструкція; двомірне зображення; алгоритм; метод; зіставлення; ітерація; матриця.

АННОТАЦИЯ

Шаповалов А.А. Технологии анализа и восстановления трехмерной графической информации на основе двумерных изображений.

Рассмотрены методы и системы обработки изображения, основные принципы трехмерной реконструкции. Проведен анализ современных методов и алгоритмов трехмерной реконструкции. Разработан алгоритм на основе метода стереосопоставления и реконструкции объекта из облака точок. Алгоритм позволяет реконструировать трехмерный объект из набора двумерных изображений с высокой точностью.

Ключевые слова: трехмерная реконструкция; двухмерное изображение; алгоритм; метод; сопоставление; итерация; комбинирование; матрица.

ANNOTATION

Shapovalov A.A. Technologies for analysis and recovery of three-dimensional graphic information based on two-dimensional images.

The methods and image processing systems, the basic principles of three-dimensional reconstruction are considered. The analysis of modern methods and algorithms of three-dimensional reconstruction. An algorithm is developed based on the method of stereo matching and reconstruction of an object from a point cloud. The algorithm allows you to reconstruct a three-dimensional object from a set of two-dimensional images with high accuracy.

Key words: three-dimensional reconstruction; two-dimensional image; algorithm; method; matching; combination; treatment; matrix.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

3D – 3 dimensional

2D – 2 dimensional

SUFR – Speeded Up Robust Features

SIFT – Scale-invariant feature transform

RGB – red, green, blue

WTA – Winner-Takes-All

GCP – Ground control points

MRF – Markov random field

SGM – Semi-Global Matching

FAST – Features from Accelerated Test

H – Висота зображення

W – Ширина зображення

BRIEF – Binary Robust Independent Elementary Features

СК – Система координат

ГСК – Головна система координат

ДСК – Допоміжна система координат

ВСТУП

В останні десятиліття обчислювальна техніка проникла в безліч сфер життєдіяльності людини. Технологічні інновації дозволяють проводити наукові дослідження, полегшують і автоматизують ручну працю, відкривають перед нами зовсім нові і привабливі можливості. Комп'ютерний зір - відносно молодий, але активно розвивається напрямок з величезною сферою застосування. Так, рішення величезної кількості завдань сучасного світу проводиться за допомогою обробки цифрових зображень. Однак для багатьох завдань вже не вистачає плоских зображень, і безліч підприємств потребують тривимірних моделях тих сцен, з якими вони працюють. Цифрові 3D моделі часто створюють вручну, коли художник чи інженер в спеціальному редакторі створює тривимірну модель. Однак цей спосіб вимагає досить багато часу і умінь. А в деяких ситуаціях потрібне отримання тривимірних моделей в режимі реального часу, і тоді навіть найспритніші людські руки не в змозі впоратися із завданням. Для роботизації та автоматизації побудови 3D моделей все частіше застосовуються методи тривимірної реконструкції - відтворення тривимірних поверхонь спостережуваних об'єктів. Реконструювання 3D моделей в умовах виробництва значно знижує тимчасові і матеріальні витрати на створення моделей, сприяє підвищенню їх точності в порівнянні з ручним моделюванням. Реконструкція тривимірної моделі сцени по набору зображень - класична задача комп'ютерного зору. Розвиток віртуальної реальності, систем управління транспортними засобами, візуалізації на основі аналізу зображень, медичної промисловості, телевізійних систем космічного призначення, а також інших областей, які потребують побудови тривимірних моделей, призвело до збільшення уваги до цієї області. Протягом останніх десятиліть системи комп'ютерного зору активно розвиваються, і на цей момент вирішено багато приватні завдання. Крім того, накопичений великий практичний і теоретичний матеріал, який полегшує розробку нових систем комп'ютерного зору. Проте, відомі рішення в ряді випадків виявляються недостатньо ефективні: як правило, для отримання точної деталізованої моделі потрібні високі обчислювальні витрати, а методи зі зниженою обчислювальною складністю не можуть забезпечити бажану точність побудови моделі.

Метою роботи є підвищення точності побудови об'ємної моделі при незначному зростанні обчислювальної складності за рахунок розробки та впровадження алгоритма тривимірної реконструкцій із хмари точок на основі зіставлення двох зображень. Для цього необхідно вирішити наступні задачі. Проаналізувати ключові характеристики методів, такі як обчислювальна складність і точність побудови моделі.

Визначені етапи аналізу:

1. Аналіз сучасних підходів до тривимірної реконструкції, методів формування об'ємних зображень.

2. Аналіз відомих алгоритмів обробки, визначення ключових характеристик методів: обчислювальна складність, точність побудови моделі.

3. Оцінка ефективності методів.

Стислий опис: Дослідження тривимірної реконструкції яка є однією з найбільш актуальних завдань комп'ютерного зору. Під тривимірної реконструкцією розуміють процес отримання цифрового уявлення тривимірної сцени реального світу - об'ємного зображення.

Важливе місце серед методів реконструкції займають підходи, пов'язані з обробкою даних, отриманих фотокамерами і відеокамерами. Завдання полягає у відновленні даних в третьому вимірі з двовимірних даних. Оскільки камера являє собою пристрій, що здійснює перетворення тривимірної сцени в двовимірне.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася в рамках науково-дослідної роботи «Система виявлення аномальних подій у відеопотоці, отриманому від множинних відеокамер».

РОЗДІЛ 1

СИСТЕМИ І МЕТОДИ ОБРОБКИ ДВОВИМІРНИХ ЗОБРАЖЕНЬ І ВІДЕОДАНИХ

У розділі проведений аналіз напрямків які займаються обробкою зображення. Так само розглянуті методи і системи обробки зображення. Розглянуті основні принципи тривимірної реконструкцій.

1.1 Напрямки обробки двовимірних зображень

Комп'ютерний зір - це технологія, за допомогою якої машини можуть знаходити, відстежувати, класифікувати та ідентифікувати об'єкти, витягуючи дані з зображень і аналізуючи отриману інформацію. Комп'ютерне зір застосовується для розпізнавання об'єктів, відеоаналітики, опису змісту зображень і відео, розпізнавання жестів і рукописного введення, тривимірної реконструкції, а також для інтелектуальної обробки зображень [1].

Машинний зір - це застосування комп'ютерного зору для промисловості і виробництва. Областю інтересу машинного зору є цифрові пристрої введення / виводу і комп'ютерні мережі, призначені для контролю виробничого обладнання.

Так виходячи з термінологій можна визначити що машинний зір використовує аналіз зображень для того, щоб вирішувати промислові завдання. Машинне і комп'ютерний зір - області пов'язані. З першого погляду може здатися, що це різні назви однієї і тієї ж технології, але це не так, тому що комп'ютерний зір - це загальна назва набору технологій, а машинне зір - сфера застосування. Так комп'ютерний зір має безліч пов'язаних наук [2].

1.2 Спеціалізовані системи обробки зображень

До спеціалізованих систем входить отримання цифрового зображення, обробка зображення з метою виділення значущої інформації на зображенні і математичний аналіз отриманих даних для вирішення поставлених завдань (рис.1.1) [3].

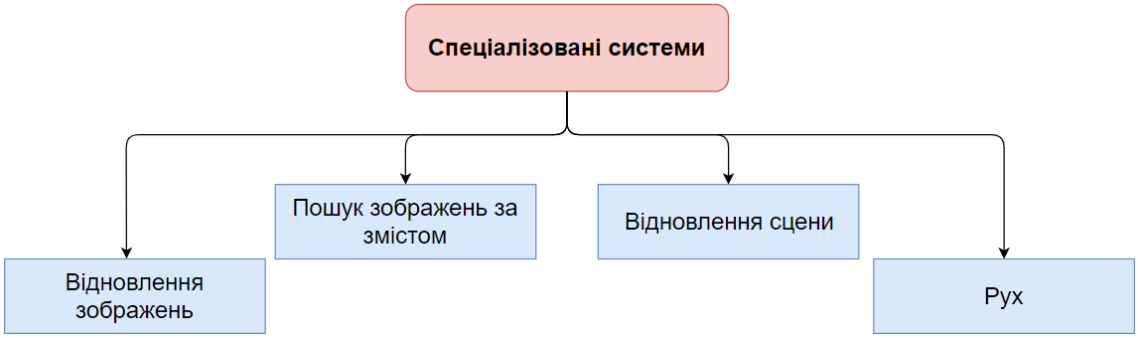


Рисунок 1.1 - Спеціалізовані системи обробки зображень

Пошук зображень за змістом - знаходження всіх зображень у великому наборі зображень, які мають певний зміст. Зміст може бути визначено різними шляхами, наприклад в термінах схожості з конкретним зображенням, або в термінах високорівневих критеріїв пошуку, що вводяться як текстові дані.

Рух - завдання пов'язані з оцінкою руху, в яких послідовність зображень (віддання) обробляються для знаходження оцінки швидкості кожної точки зображення або тривимірної сцени. А саме завдання таких систем полягає в:

1) Визначення тривимірного руху камери.
2) Стеження, тобто слідування за переміщеннями.

Відновлення сцени - відновлення сцени з двох або більше зображення сцени, або відеоданих. Відновлення сцени має завданням відтворити тривимірну модель сцени. У найпростішому випадку, моделлю може бути набір точок тривимірного простору. Більш складні методи відтворюють повну тривимірну модель.

Відновлення зображень - Завдання відновлення зображень це видалення шуму (шум датчика, розмитість рухомого об'єкту). Найбільш простим підходом до вирішення цього завдання є різні типи фільтрів, таких як фільтри нижніх або середніх частот. Більш складні методи використовують уявлення того, як повинні виглядати ті чи інші ділянки зображення, і на основі цього їх зміна.

Більш високий рівень видалення шумів досягається в ході первинного аналізу відеоданих на наявність різних структур, таких як лінії або межі, а потім управління процесом фільтрації на основі цих даних [4].

1.3 Методи обробки зображення

У системах обробки зображення, для вирішення перерахованих завдань, використовуються різні технології і методи [5].

Лічильник пікселів: підраховує кількість світлих або темних пікселів і на основі результату робить необхідні висновки про зображення.

Виділення пов'язаних областей: Зв'язкова область зображення - це, з одного боку, тип об'єкта, все ще дуже близько пов'язаний з растровим зображенням, і в той же час - це вже якась самостійна семантична одиниця, що дозволяє вести подальший геометричний, логічний, топологічний і будь-який інший аналіз зображення.

Бінаризація: перетворює зображення в сірих тонах в бінарне (білі і чорні пікселі).

Гістограма і гістограмна обробка: гістограма характеризує частоту зустрічальності на зображенні пікселів однакової яскравості.

Сегментація: використовується для пошуку або підрахунку деталей. Сегментацією зображення називається розбиття зображення на несхожі за деякою ознакою області. Передбачається, що області відповідають реальним об'єктам, або їх частин, а кордони областей відповідають кордонам об'єктів.

Оптичне розпізнавання символів: автоматизоване читання тексту, наприклад, серійних номерів.

Вимірювання: вимірювання розмірів об'єктів в дюймах або міліметрах.

Зіставлення шаблонів: пошук, підбір, і підрахунок конкретних моделей.

Інваріантні алгоритми зіставлення точкових особливостей на зображеннях: виявлення і зіставлення точкових особливостей на зображеннях.

Різні методи відновлення форми об'єкта по зображеннях.

У більшості випадків, системи комп'ютерного зору використовують послідовне поєднання цих методів обробки для виконання повного інспектування [5].

1.4 Компоненти системи

Типова система обробки зображення складається з однієї або декількох цифрових або аналогових камер (чорно-білі або кольорові) з відповідною оптикою для отримання зображень, підсвічування і об'єкта, обладнання введення / виведення або канали зв'язку для доповіді про отримані результати. Крім того, важлива і програмна складова систем обробки зображення, а саме програмне забезпечення для підготовки зображень до обробки (для аналогових камер це оцифровщик зображень), специфічні додатки програмного забезпечення для обробки зображень і виявлення відповідних властивостей (рис.1.2) [6].

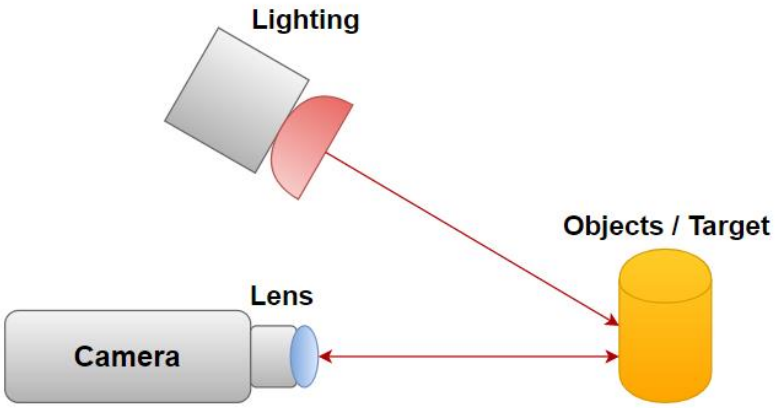


Рисунок 1.2 - Склад типової системи машинного зору

Матриця чутливих елементів, що входять до складу відеокамери, призначена для отримання цифрового зображення. До складу матриці чутливого елемента входить безліч аналого-цифрових перетворювачів, призначених для перетворення інформації про світловий інтенсивності в цифрове значення [6].

Об'єктив дозволяє камері фокусуватися на певній відстані і отримувати чітке зображення об'єкта. У разі, коли об'єкт знаходиться поза фокусної відстані, зображення виходить розмитим (розмитим, з нечіткими краями), що погіршує можливість обробки відеоряду. На відміну від звичайних цифрових фотоапаратів з об'єктивами, що підтримують функції автофокусування, в комп'ютерному зорі застосовується оптика з фіксованою фокусною відстанню або налаштуванням фокусу. Існують різні типи об'єктивів для самих різних завдань (стандартні,

телескопічні, з широким кутом огляду, зі збільшенням та інші), і вибір правильного типу оптики - важливий етап при проектуванні системи комп'ютерного зору [6].

Підсвічування - ще один важливий елемент в комп'ютерному зорі. Завдяки використанню різних типів освітлення можна розширити коло завдань, що вирішуються комп'ютерним зором. Існує різні типи підсвічування, але найбільш популярним є світлодіодна - в зв'язку з її високою яскравістю. При цьому сучасний рівень розвитку світлодіодної техніки забезпечує великий термін служби пристрою і мале енергоспоживання [6].

1.5 Принципи функціонування систем обробки зображення

Реалізація систем обробки зображення сильно залежить від сфери їх застосування, апаратної платформи і вимог по продуктивності. Деякі системи є автономними і вирішують специфічні проблеми детектування і вимірювання, тоді як інші системи складають підсистеми більших систем, які вже можуть містити підсистеми контролю механічних маніпуляторів, інформаційні бази даних, інтерфейси людина-машина (рис.1.3) [4] [6] [7].

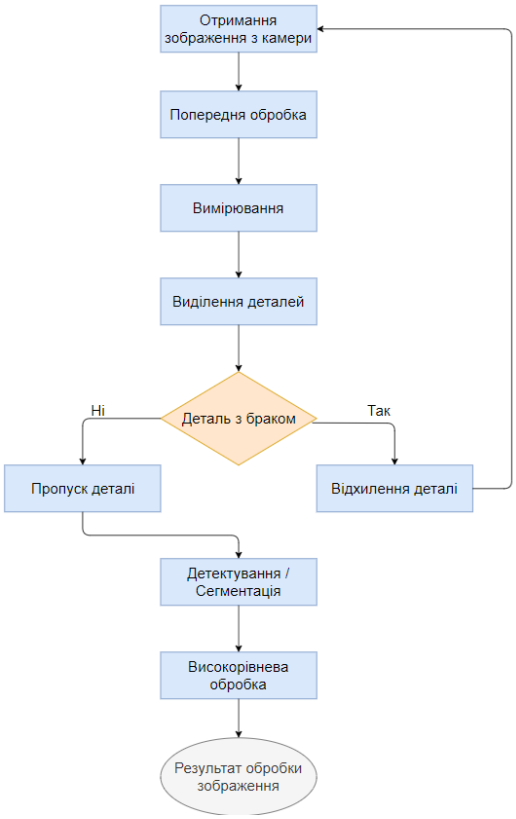


Рисунок 1.3 - Послідовність дій системи комп'ютерного зору

Отримання зображень**:** цифрові зображення виходять від одного або декількох датчиків зображення. Залежно від типу датчика, що виходять дані можуть бути звичайним двомірним зображенням, тривимірним зображенням або послідовністю зображень. Значення пікселів зазвичай відповідають інтенсивності світла в одній або декількох спектральних смугах (кольорові або зображення у відтінках сірого), але можуть бути пов'язані з різними фізичними вимірами, такими як глибина, поглинання або відбиття звукових або електромагнітних хвиль.

Попередня обробка: перед тим, як методи комп'ютерного зору можуть бути застосовані до відеоданих з тим, щоб витягти певну частку інформації, необхідно обробити відеодані, з тим щоб вони задовольняли деяким умовам, в залежності від використовуваного методу. Прикладами є:

1) повторна вибірка з тим, щоб переконатися, що координатна система зображення вірна;
2) видалення шуму з тим, щоб видалити спотворення, що вносяться датчиком;
3) поліпшення контрастності, для того, щоб потрібна інформація могла бути виявлена;
4) масштабування для кращого розрізнення структур на зображенні.

Вимірювання: визначає об'єкта, розміри, дефекти та інші характеристики зображення.

Виділення деталей: деталі зображення різного рівня складності виділяються з відеоданих. Типовими прикладами таких деталей є:

1) лінії, кордони і кромки;
2) локалізовані точки інтересу, такі як кути, краплі або точки: більш складні деталі можуть ставитися до структури, формі або руху.

Якщо деталь йде з браком, програмне забезпечення подає сигнал пристрою для відхилення деталі. Де потрібно повторити алгоритм попередньої обробки зображення, або отримати нове зображення деталі.

Детектування / Сегментація: на певному етапі обробки приймається рішення про те, які точки або ділянки зображення є важливими для подальшої обробки. Прикладами є:

1) виділення певного набору точок;
2) сегментація одного або декількох ділянок зображення, які містять характерний об'єкт.

Високорівнева обробка: на цьому етапі вхідні дані зазвичай представляють невеликий набір даних, наприклад набір точок або ділянку зображення, в якому ймовірно перебуває певний об'єкт [4] [6] [7]. Прикладами є:

1) перевірка того, що дані задовольняють умовам, що залежать від методу і застосування;
2) оцінка характерних параметрів, таких як положення або розмір об'єкта;
3) класифікація виявленого об'єкту за різними категоріями.

1.6 Тривимірна реконструкція

Під тривимірної реконструкцією розуміють процес отримання цифрового уявлення тривимірної сцени реального світу - об'ємного зображення. Важливе місце серед методів реконструкції займають підходи, пов'язані з обробкою даних, отриманих фотокамерами і відеокамерами. Завдання полягає у відновленні даних в третьому вимірі з двовимірних даних [8]. Оскільки камера являє собою пристрій, що здійснює перетворення тривимірної сцени в двовимірне зображення, при переході неминуче втрачається інформація про кути, справжні розміри і т.д. [9]. Існуючі підходи можна розділити на активні та пасивні. В активних джерело випромінювання і приймач синхронізовані деяким чином, в той час як в пасивних - не синхронізовані [9].

На етапі збору даних потрібно отримати дані про дальність для деякого набору різних видів, які містять всю поверхню об'єкта. Число таких видів залежить від поставленого завдання, вимог, що пред'являються до точності і детальності моделі. Так, часто досить 810 видів, однак для складних об'єктів і для забезпечення високої точності необхідна більша кількість видів. Важливо розуміти, що з ростом числа видів зростає і обчислювальна складність методу тривимірної реконструкції [9].

Кожен отриманий вид складається з одного дальнометріческого зображення частини поверхні об'єкта і часто ще з полутонового або кольорового зображення. Дальнометріческіе дані від всіх видів повинні бути скомбіновані для отримання моделі поверхні об'єкту. Дані інтенсивності можуть бути використані в якості текстур для подальшого реалістичного відображення в графічних додатках.

Процес комбінування дальнометріческіх даних шляхом приведення їх до єдиної тривимірної системі координат називається суміщенням (реєстрацією). Існуючі підходи в тривимірної реконструкції доцільно розділити на класи відповідно до інформації, використовуваної для оцінки тривимірної форми сцени:

1. методи, які залежать від геометрії;
2. методи, які використовують фотометричну інформацію [9].

Використання інформації про геометрію ефективно в разі відносно простих об'єктів сцени: сфер, кубів і т.д. У той же час, фотометрична інформація дозволяє оцінювати тривимірну форму в більш складних випадках [9].

1.7 Класифікація методів тривимірної реконструкцій

Відомо безліч методів тривимірної реконструкції. Розглянемо основні методи тривимірної реконструкції [3]:

1) стерео зір;
2) часпрогонові сенсори;
3) структурований світло;
4) структура з руху;
5) структура з освітленості;
6) структура з затінення;
7) глибина з фокусування / расфокусировки (shape-from-focus).

Одним з популярних підходів, є стерео зір. Бінокулярний стерео зір, на перший погляд, видається найбільш інтуїтивно зрозумілим. Точки поверхні предмета дають зображення, відносне положення яких залежить від відстані до точки спостереження [10].

Для реалізації цього підходу потрібні дві або більше камери, спрямовані на сцену, попередньо відкалібровані. Так як області зору камер в стереопарі перетинаються, можуть бути знайдені відповідності одним і тим же частинах сцени на зображеннях стереопари. Ефект обсягу виникає в силу того, що розташовані на різній відстані від спостерігача частини сюжету при перегляді з різних точок мають різне кутовий зсув, зване параллаксом [11].

При цьому параллактическое зміщення об'єкта сцени буде тим більше, чим ближче він розташований до камери. Знаючи зміщення для кожної точки сцени і параметри калібрування стереопари можна отримати так звану карту глибини. Карта глибини - зображення, що містить далекомірну інформацію. Значення кожного пікселя цього зображення відповідає відстані від площини сенсора до об'єкта сцени. Проектування точки X на картинні площині двох камер - xl і xr (рис.1.4).

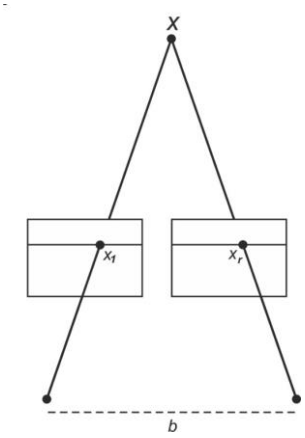


Рисунок 1.4 - Проектування точки зйомки на площині зображень камер стереопари [9]

Надійність системи отримання зображень можна підвищити шляхом збільшення кількості камер. Одна камера буде базовою, в її системі координат буде обчислюватися дальнометріческое зображення. Точка на поверхні об'єкта повинна бути видима базової камерою, проектором і, як мінімум, ще однією камерою. Тоді процес обробки зводиться до звичайної обробці в стереосистеми двома камерами. Якщо точка видима відразу трьома або більше камерами, включаючи базову, то додаткові зображення можна використовувати для підвищення надійності обробки [9].

Часопролєтні сенсори аналізують орієнтацію і положення об'єктів в полі зору камери, і розраховують відстань до них від камери. Так часопролєтні сенсори вимірюють час прольоту світлового сигналу, що випускається камерою, і відбитого кожною точкою одержуваного зображення. За принципом роботи такі системи нагадують радари або сонари.

Пристрій безперервно посилає ультракороткі лазерні імпульси, які, відбиваючись, повертаються назад, і реєструються за допомогою сенсора. Для відомої швидкості поширення імпульсів, використовуючи точну електронну схему для вимірювання точного часу передачі і прийому сигналу, відстань D можна визначити як [12]:

$$D = \frac{\Delta\varphi}{2\pi f} \tag{1.1}$$

Де с - являє собою швидкість поширення проміння (наприклад, швидкість світла для лазерного сенсора), а Δφ - різниця фаз між відправленим і прийнятим відбитим сигналом, f – частота (рис.1.5) [12].

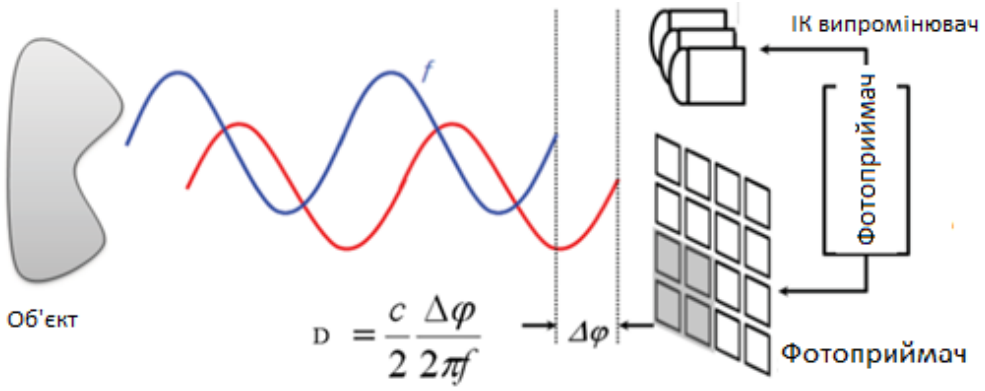


Рисунок 1.5 - Принцип роботи часопролетних сенсорів [24]

Також можливе додаткове вимірювання фонового випромінювання, що допомагає зробити тривимірну картинку ще більш точною. Спочатку надсилається перший імпульс, потім, ще до його закінчення, надсилається другій імпульс такої ж тривалості. Перший вимір полягає в накопиченні відбитого сигналу від першого імпульсу. Друге - в накопиченні відбитого сигналу за куди більш тривалий проміжок часу. Після вирахування фонового випромінювання (третій вимір) за величиною цих імпульсів визначається відстань до об'єкта [9].

Метод структурованого світла. Поверхні об'єктів висвітлюються через шаблон у вигляді сітки: в даному випадку це реалізується за допомогою слайд-проектора, що проектує регулярну сітку зі світлих ліній на поверхні об'єктів. Камера при формуванні зображень передає результат спотворення сітки за рахунок форми і орієнтації поверхні. Так як структура прямокутної сітки добре відома, то система має апріорну інформацію про те, які саме проектують промені формують точки зображення, відповідні перетину ліній сітки (рис.1.6).

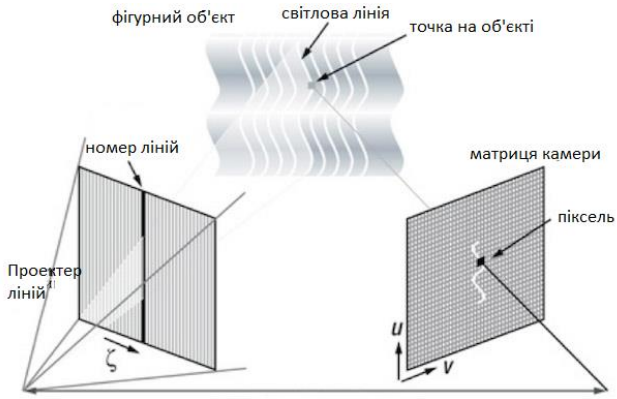


Рисунок 1.6 - Структурований світ [25]

Глибина з фокусування / расфокусировки - завдання визначення тривимірної сцени по набору з двох або більше зображень. Зображення виходять в результаті зміни параметрів камери і зйомки з тієї ж позиції (рис.1.7) [13].

Різниця між глибиною з фокусування і глибиною з расфокусировки полягає в тому, що в першому випадку можна динамічно змінювати параметри камери в процесі оцінки поверхні, чого не можна робити в другому [14] [15]. Обидва методи можна віднести як до пасивних, так і до активних, в залежності від того, чи можна використовувати структурований світло.

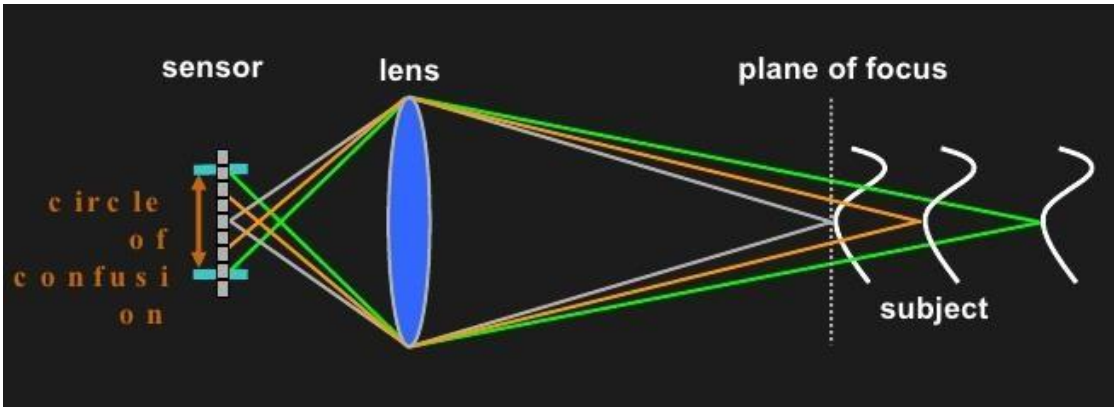


Рисунок 1.7 - Принцип роботи методу глибина з фокусування та расфокусировки [26]

Структура з руху використовується для знаходження тривимірної структури об'єкта шляхом аналізу локального руху частин сцени з плином часу. Коли переміщається приймач (камера) або об'єкт, або вони обидва, то камера реєструє інформацію з послідовності змінюються зображень (рис.1.8) [17]. За векторах оптичного потоку або відповідних точок на тривимірних

сценах можна відновити поверхні і кути, а також визначити траєкторію руху сенсора через спостережувану сцену. Цей інтуїтивно зрозумілий процес можна формалізувати і отримати цілий клас специфічних математичних задач.

Знаходження структури з руху являє собою задачу, аналогічну завданню стерео зору (рис.1.8). В обох випадках потрібно знайти відповідність між зображеннями. Тільки в разі структури з руху зображення отримані в різний час. Для знаходження відповідності часто використовуються детектори особливих точок, а також дескриптори локальних областей, такі як SIFT [18] або SURF [19].

Система дозволяє визначати не тільки тривимірну структуру об'єкта, але також і параметри його руху щодо системи координат камери.

Отримання тривимірних координат і пошук відповідників даною групою методів схильний до помилок. Алгоритми, засновані на відповідність характерних ознак, здатні працювати з великими часовими проміжками, але обчислюють тільки розріджені тривимірні структури. Алгоритми, засновані на оптичному потоці, використовують невеликі тимчасові проміжки між зображеннями і обчислюють щільні тривимірні структури [9].

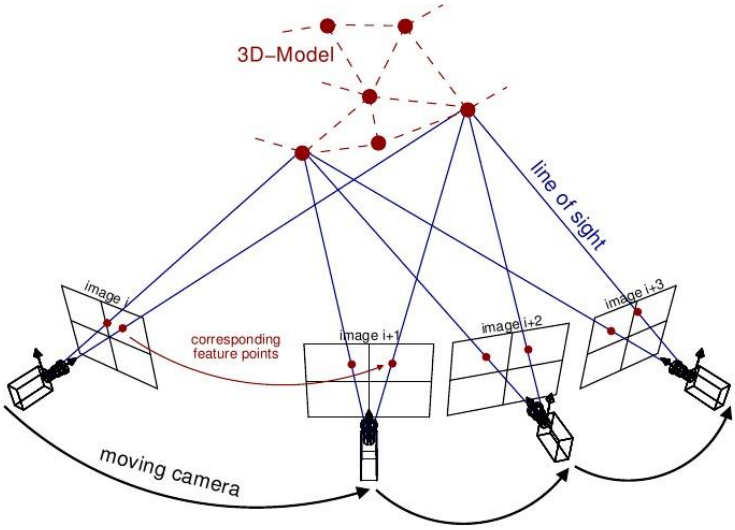


Рисунок 1.8 - Знаходження структури з руху [17]

Метод оцінки форми по освітленості заснований на особливостях роботи зорового аналізатора людини. Спостерігач схильний вважати плавно затінені поверхні віддаляються від напрямку погляду.

За допомогою формули відображення по закону Ламберта можна пов'язати інтенсивність (освітленість) елемента поверхні і напрям нормалі до поверхні в цьому місці. Метод для обчислення форми по освітленості поверхні обчислює нормаль до точки поверхні N = f (x, y) за даними про інтенсивність її освітленості, отриманим з зображення поверхні. N - це нормаль до точки поверхні, що відповідає точці зображення (x, y) (рис.1.9) [9].

Зліва показано зображення об'єктів, добре описуються законом відображення Ламберта: інтенсивність освітленості на зображенні пропорційна куту між нормаллю до поверхні і напрямком падаючого світла. Справа показані нормалі до поверхні в декількох точках на поверхні об'єктів. Добре видно, що більш яскраві точки зображення відповідають елементам поверхні, нормалізує обмін речовин до яких спрямовані прямо на джерело світла: нормалі до поверхні, спрямовані «прямо на спостерігача», на (рис.1.9) праворуч відзначені хрестиками. Нормалі до поверхні перпендикулярні і до напрямку спостереження, і до кордону поверхні на зображенні в точках лімба. Ці умови строго обмежують нормаль в тривимірному просторі. Використовуючи ці обмеження, можна поширити вектори нормалі на всі крапки поверхні. Таким чином, можна побудувати часткове внутрішнє зображення. Для отримання глибини z в кожній точці зображення ми можемо привласнити довільне значення $z0$ однією з найбільш яскравих точок і потім поширити значення глибини вздовж зображення, з огляду на зміну напрямків нормалей.

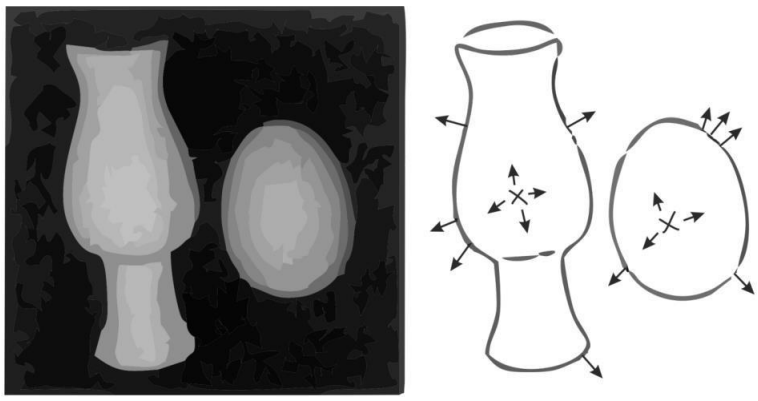


Рисунок 1.9 - Форми по освітленості [9]

Тіні на зображеннях, отриманих з різних ракурсів або при різних умовах освітлення, можуть містити значну кількість інформації про форму поверхні. Значення яскравості пікселів можуть бути використані для визначення орієнтації поверхні в локальній околиці. При використанні інформації з багатьох пікселів може бути отримана карта глибини поверхні [9].

Інформація про поляризацію світла, відбитого від поверхні, дає можливість визначення орієнтації поверхні. Звичайні системи отримання зображень можуть бути використані тільки для запису інтенсивності світла, відбитого від об'єкта спостереження. Даний методи є вузько спрямованим і використовуються в медичній томографії. Таким чином, деякі біологічні тканини і матеріали будуть виглядати однорідними, навіть якщо вони мають якусь внутрішню структуру. У деяких тканинах, таких як м'язи, сухожилля, людська рогівка і сітківка, оцінка стану цих структур може бути цінним інструментом для визначення загального стану здоров'я. Сітківка людського ока, наприклад, складається з декількох шарів різних типів клітин, через які світло повинне проходити до досягнення фоторецепторів шару. Дані про поляризацію відбитого світла можуть бути використані для кращої реконструкції форми поверхні [9].

1.8 Подання тривимірних моделей

Суміщення дальнометріческіх даних і приведення їх до єдиної системи координат може слідувати етап побудови тривимірної моделі. Як правило, важливо, щоб реконструйований об'єкт за формою максимально близько був схожий на реальний спостережуваний об'єкт і передавав його топологію [9]. Так було проведено аналіз останніх досліджень тривимірної реконструкції (табл.1.1.). Та проведено аналіз моделей, що використовуються в задачах тривимірної реконструкції (табл.1.2). На сьогоднішній день відомий ряд подань для більш точного опису об'єктів з криволінійними і довільними поверхнями. До них відносяться:

a) моделі на основі полігональних сіток;
b) моделі типу грань-ребро-вершина;
c) моделі на основі точкових уявлень (хмари точок);
d) воксельні моделі і моделі на основі октантних дерев; моделі, засновані на картах глибини.

Полігональні сітки (подання у вигляді списку граней) є найбільш поширеним уявленням, для якого є велика кількість програмних продуктів для редагування і візуалізації об'єктів. Важливою особливістю моделей на основі полігональних сіток є зв'язність, яка спрощує генерацію штучних поверхонь.

Моделі типу грань-ребро-вершина використовуються в задачах, де потрібно швидкий доступ до всіх елементів уявлення тривимірного об'єкту. Модель дозволяє уявити об'єкт у вигляді безлічі граней, безлічі ребер і безлічі вершин, в такому поданні вони задані явно.

Воксельні моделі представляються у вигляді регулярних тривимірних масивів, елементів яких зіставляють характеристики кольору і прозорості [20], [21]. У зв'язку з тим, що в воксельного поданні зберігається опис всієї області простору, що містить в собі тривимірний об'єкт, то обсяги даних в воксельних уявленнях значні навіть для невеликих моделей. Для зменшення обсягу уявлення найбільш часто використовуються деревовидні ієрархії. Воксельні уявлення орієнтовані на візуалізацію технічних об'єктів, віртуальну реальність, зберігання медичних даних, а також на інші області.

У точкових уявленнях об'єкти моделюється як набір тривимірних координат точок, що належать поверхням об'єкта. Основною проблемою таких уявлень є відсутність даних про зв'язності точок і їх приналежності до безперервним поверхонь сканованого тривимірного об'єкту. У зв'язку з неоднозначністю вирішення завдання відновлення безперервної поверхні можлива поява розривів [23].

Моделі, засновані на картах глибини [24] не використовують проміжні структури даних, і представляють тривимірний об'єкт за допомогою двовимірних зображень, доповнених компонентом, визначальним глибину для кожної точки. Базовою структурою даних є набір зображень з картами глибини. Пара «зображення - карта глибини» визначається як RGB-зображення, яким співставлено зображення grayscale того ж розміру, значення кожної точки якого визначається відстанню від камери до поверхні об'єкту [24].

Таблиця 1.1 - Аналіз останніх публікацій тривимірної реконструкції

Дослідження	**Завдання**	**Використовувані методи**	**Результати**	**Основні результати та висновки**	**Коментарі**
1	**2**	**3**	**4**	**5**	**6**
Company C3D Labs, (2019)	Конвертація полігональної сітки в B-rep модель.	Сегментація полігональної моделі.	Програмний компонент C3D B-Shaper.	Метод триступеневої конвертації полігональної моделі.	У даній роботі пропонується метод і програмне забезпечення яке дозволить перетворювати полігональні моделі в граничні уявлення.

1	2	3	4	5	6
Michael Himpel; André Melzer (2019)	Three-Dimensional Reconstruction of Individual Particles in Dense Dust Clouds: Benchmarking Camera Orientations and Reconstruction Algorithms	This paper benchmarks two approaches capable of reconstructing the trajectories of particles in three dimensions. The influences of the particle number, the particle number density, and the orientation of the individual cameras are studied.	Three-dimensional algorithm for reconstructing individual particles in dense dust clouds	На основі методу мульти-тріангуляції	Даний алгоритм визначає тривимірне положення об'єкта в пиловий плазмі.
Фасхітдінв Р.Р. (2015)	Сегментація і реконструкція полігональної моделі по знімках комп'ютерної томографії нижніх кінцівок.	Воксельна модель; методи сегментації.	Програмний модуль автоматичної сегментації	Підхід комбінування методів.	В рамках даної роботи пропонується: програмний модуль для створення полігональної моделі по набору томографічних знімків.
Реут А.В. (2016)	Алгоритм візуалізації форми просторового об'єкта, поданого його матричною моделлю.	Воксельна модель.	Алгоритм візуалізації просторового тіла по його дискретної матричної моделі.	Побудова проміжної воксельної моделі поверхні тіла і визначення на її основі полігональної сітки з трикутними гранями.	В рамках даної роботи пропонується алгоритм який дозволяє зробити візуалізацію поверхні, що зменшує обсяг проміжних даних, усуваються неоднозначності і скорочується процес обчислень.

1	2	3	4	5	6
Yue Zhao; Lei Zhang; Bo Jiu; Hongwei Liu; Zhenfang li (2019)	Practical fully three-dimensional reconstruction algorithms for diffuse optical tomography	Based on projective transformation method and epipolar geometry	Three-dimensional object of a cosmic body	The algorithm builds the projection equations between the target 3D geometry and ISAR images without cross-range scaling constraint.	У цій статті пропонується алгоритм тривимірної реконструкції для космічних цілей з мультістатическ імі системами ISAR.
Алсинбаев К.С. (2015)	Алгоритм визначення тіл об'ємних об'єктів в тривимірному нерегулярном у хмарі точок.	Алгоритм формування фігур на основі розпізнавання тіла; алгоритми Брезхейма; Кластеризація точок на регулярній мережі.	Алгоритм кластеризації хмари точок.	Алгоритм здійснює на виключення з порівняння точок, що знаходяться на досить великій відстані від поточної яка перевіряєть ся.	Основним завданням в роботі є модифікацією алгоритму (Кластеризація точок на регулярній мережі) для підвищення обчислювальної точності.
Darae Jeong; Yibao Li; Heon Ju Lee; Sang Min Lee; Junxiang Yang; Seungwoo Park; Hyundong Kim; Junseok Kim (2017)	Efficient 3D Volume Reconstruction from a Point Cloud Using a Phase-Field Method	Phase field method and on on the method of segmentation of a three-dimensional binary image..	Volume reconstruction from unorganized point clouds	The algorithm reconstructs a three-dimensional object from a point cloud based on the binary image segmentation method	У цій статті розроблений гібридний чисельний алгоритм для ефективної реконструкції тривимірної моделі з неорганізовани х хмар точок.
Вітіска Н.І. Гуляєв Н.А. (2016)	Метод візуалізацій тривимірної сцени і об'єктів воксельної графіки для систем імітаційного моделювання.	Воксельні моделі і моделі на основі октантних дерев.	Метод організації вмісту сцени, нерівномірно го поділу з розбивкою простору всередині обсягу.	Метод заснований на трасуванні променів.	Метод використовує трасування променів візуалізацій до проведення операцій з одиничним вокселя.

1	2	3	4	5	6
Красильников Н.Н. (2015)	Методи конвертації двовимірних зображень і відео в стереоскопічних формат.	Методи конвертації.	Метод конвертації, заснований на використанні фактора руху при формуванні карти глибини	Метод конвертації двовимірних зображень в тривимірне зображення, заснований на примітивній карті глибини.	У даній роботі пропонується метод побудова карти глибині по набору кадрів з відео даних.

Таблиця 1.2 - Аналіз моделей, що використовуються в задачах тривимірної реконструкції

Моделі	Папери, де були використані ці моделі	Застосування моделей в задачах реконструкції	Плюси	Мінуси
1	2	3	4	5
моделі на основі полігональних сіток	Company C3D Labs, (2019); Фасхітдінв Р.Р. (2015); Кулешов С.В. (2015); Гонахчян В.И. (2016).	Тривимірне сканування; моделювання за допомогою примітивів; реконструкція певного об'єкта зі сцени.	Висока точність моделі; набір інструментів побудови моделі; редагування моделі на етапі реконструкцій.	Використання технічного обладнання; високо обчислювальна складність; розпізнавання поверхонь різного типу.
моделі типу грань-ребро-вершина;	Реут А.В. (2016); Гаранжі В.А., Кудрявцева Л.М (2012)	Математичне моделювання; побудова геометричних об'єктів.	Не висока складність обробки; висока обчислювальна швидкість реконструкції.	Втрата точності реконструкції об'єкта; реконструкція простих форм об'єкта.
моделі на основі точкових уявлень (хмари точок);	Алсинбаев К.С. (2015); Аксьонов А.Ю. (2015) Юркін В.А. (2016). Шаповалов Р.В. (2010)	Моделювання ландшафту в ГІС системах; реконструкція об'єктів в руху, моделювання об'єктів з томографічних зображень; лазерне сканування.	Висока точність моделі; не висока складність обробки; розпізнавання тіла об'єкта.	Втрата обсягу структури тіла об'єкта; розриви між хмарами точок.

1	2	3	4	5
воксельні моделі і моделі на основі октантних дерев;	Арсланов В.І. (2016); Вітіска Н.І. Гуляєв Н.А. (2016)	Реконструкція певного об'єкта зі сцени; математичне моделювання; побудова геометричних об'єктів.	Необмежений рівень деталізації; висока точність моделі.	Використання технічного обладнання; високо обчислювальна складність; статичність об'єкта; недолік оптимальних алгоритмів
моделі, засновані на картах глибини	Красильников Н.Н. (2015); Воронін В.В. (2014 року); Зачосом А. (2015).	Реконструкція кадрів відеоданих; Повна реконструкція сцени яка міститься в зображень	Реконструкція сцени; висока точність реконструкцій об'єктів на сцені; збереження розмірів об'єктів.	Недостатній рівень освітленості об'єкта; недостатній рівень надійності систем; велика залежність точності ідентифікації від положення об'єкта.

Аналіз останніх публікацій тривимірних реконструкцій показав що методи і алгоритми мають свої плюси і мінуси. Залежно від поставленого завдання використовується той чи інший метод реконструкцій. Для більш детальної реконструкції тривимірної моделі у багатьох методах і алгоритмах, потрібна попередня обробка зображення: видалення фону; настройка світла; отрегуліровка камери; видалення шумів. Так деякі методи і алгоритми мають сильні сторони такі як висока точність побудова моделі, але мають високу обчислювальну складність. Для підвищення точності тривимірної реконструкції з найменшим зростання обчислювальної складністю, методи і алгоритми можуть бути комбіновані, таким прикладом є алгоритм "Practical fully three-dimensional reconstruction algorithms for diffuse optical tomography". Виходячи з аналізу, можна зробити висновок, що основні завдання даної магістерської роботи будуть:

1) Проведення аналізу методів та алгоритмів реконструкції об'ємної моделі.
2) Розробка алгоритму з комбінуванням методів.
3) Програмна реалізація на мові C ++ з використанням бібліотеки OpenGL.
4) Проведення експериментів на основі запропонованого алгоритму.

1.9 Висновки до першого розділу

Комп'ютерний зір активно розвиватися, і є актуальним напрямком дослідження в наш час. Комп'ютерний зір спрямований на обробку інформацій отриманого з двомірних або тривимірних зображень, і в залежності від поставленого завдання, дозволяє отримати певний результат. У свою чергу комп'ютерний зір має ряд суміжних наук і дисциплін, і формують пильну увагу представників з різних наукових співтовариств до даної галузі. Так комп'ютерний зір застосовується в різних сферах сучасного світу, таких як медицина, системи охорони, системи виявлення і збору інформацій, торгівлі і.т.д. Різні методи обробки зображень впроваджуються в перераховані вище сфери, і допомагають автоматизувати роботу різних систем, а так само зменшити кількість витратних ресурсів і часу. Так було проведено аналіз систем і методів обробки зображень. Розглянуто основні інструменти для обробки зображень.

Одним з найпопулярніших напрямків комп'ютерного зору є реконструкція тривимірних сцен і об'єктів з двомірних зображень. У цьому розділі були виявлені ключові моменти тривимірної реконструкції, і була проведена класифікація методів тривимірної реконструкції. Так визначено що розроблено безліч методів і алгоритмів, які дозволяють побудувати тривимірну модель. Але багато методів і алгоритмів реконструкцій мають ряд проблем, а саме точність побудова моделі, розпізнавання об'єкта, алгоритми обробки зображення комп'ютерного зору, котрі не завжди дозволяють реконструювати об'єкти точним чином. Виходячи з висновка потрібно провести детальний аналіз методів та алгоритмів побудови об'ємної моделі.

1.10 Перелік джерел посилань до розділу 1

1) Візільтер Ю. В., Желтов С. Ю., Князь В. А., Ходарев А. Н., Моржін А. В. Обробка та аналіз цифрових зображень з прикладами на LabVIEW IMAQVision 2007.

2) Бобровський С. «Коли машини прозріють». Посилання: https://www.itweek.ru/themes/detail.php?ID=66663&sphrase_id=12198.

3) А. Т. Вахитов, Л. С. Гуревич, Д. В. Павленко. Огляд алгоритмів стерео зору. Санкт-Петербурзький державний університет.

4) Wikipedia. Комп'ютерне зір. Типові завдання комп'ютерного зору. Посилання: https://ru.wikipedia.org/wiki/Компьютерное_зрение.

5) Технічна складова машинного зору. Посилання: https://studbooks.net/2175437/informatika/tehnicheskaya_sostavlyayuschaya_mashinnogo_zreniya.

6) І.Г Благовіщенський. Використання систем комп'ютерного зору для контролю в режимі онлайн якості сировини і готової продукції харчової промисловості.

7) А.А. Лукьяница, А.Г. Шишкин. Цифровая обработка видеоизображений 2009. С 518.

8) Чочиа П.А. Переход от 2D к 3D-изображениям: модификация двух-масштабной модели и алгоритмов обработки. Посилання: https://cyberleninka.ru/article/v/trehmernye-i-dvumernye-izobrazheniya-modeli-algoritmy-i-oblasti-analiza.

9) Науково-освітній курс «Завдання зіставлення локальних точок в 3D реконструкції». Посилання: https://docplayer.ru/58534378-Nauchno-obrazovatelnyy-kurs-zadacha-sopostavleniya-lokalnyh-tochek-v-3d-rekonstrukcii.html.

10) Schlesinger M.I., Flach B. Some solvable subclass of structural recognition problems. CzechPatternRecognitionWorkshop 2000. pages 55-62.

11) Хорн Б. К. П. Зір роботів: Пер. з англ. М .: Мир, 1989. С 487.

12) Shim H., Lee S. Performance evaluation of time-of-flight and structured light depth sensors in radiometric / geometric variations // Optical Engineer- ing. 2012 vol. 51, issue 9.

13) Favaro, P. Depth from focus/defocus. Посилання: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FAVARO1/dfdtutorial.html.

14) Chaudhuri S., Rajagopalan A. Depth from defocus: a real aperture imaging approach. Springer Verlag, 1999.

15) Ens J., Lawrence P. An investigation of methods for determining depth from focus. // IEEE Trans. Pattern Anal. Mach. Intell, 1993, pages 97-108.

16) Y. Xiong and S. Shafer. Depth from focusing and defocusing. In Proc. of the Intl. Conf. of Comp. Vision and Pat. Recogn, 1993, pages 68-73.

17) Exploring the use of 3D GIS as an analytical tool in archaeological excavation practice. Посилання: https://www.researchgate.net/publication/303824023_Exploring_the_use_of_3D_GIS_as_an_analytical_tool_in_archaeological_excavation_practice.

18) Lowe D. G. Distinctive Image Features from Scale-Invariant Keypoints. Посилання: https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf.

19) Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pages 346—359.

20) Laur D., Hanrahan P. Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering. / Proc. SIGGRAPH 1991.

21) Westover L. Footprint Evaluation for Volume Rendering. / Proc. SIGGRAPH'90.

22) Levoy M., Whitted T. The Use of Points as a Display Primitive: Technical Report / TR 85-022, University of North Carolina at Chapel Hill, 1985.

23) Debevec, P. Introduction to Image-Based Modeling, Rendering, and Light- ing. / P. Debevec // SIGGRAPH'2000 courses.

24) Vijaya Kumar Ghorpade. 3D Semantic SLAM of Indoor Environment with Single.

25) Структурований світло в Kinect. Посилання: http://robotosha.ru/robotics/structured-light-kinect.html.

26) Shape from focus system - Computer Vision and Pattern Recognition, 1992.

РОЗДІЛ 2

АНАЛІЗ МЕТОДІВ І АЛГОРИТМІВ ПОБУДОВИ ОБ'ЄМНОЇ МОДЕЛІ

У другому розділі проведено аналіз сучасних методів і алгоритмів. А саме: методи ректіфекацій; методи пошуку пов'язаних точок; методи фокусування і расфокусировки; методи пошуку ключових точок; методи побудовах тривимірної моделей з хмар точок.

2.1 Методи ректифікації

2.1.1 Метод Хартлі

Одним з найбільш популярних методів ректифікації є алгоритм, запропонований Хартлі [1]:

1) Визначення точкових відповідностей між зображеннями. Можуть використовуватися будь-які з відомих методів виділення характерних точок і метрики.

2) Визначення фундаментальної матриці F на основі цих відповідностей.

3) Вибір проективного перетворення $H' = G$, яке відображає епіполюс p' на нескінченність (в точку (1,0,0)) для епіполюса з координатами $(f,0,1)$ і $H' = GRT$ для координат в загальному вигляді, де перетворення R і T - поворот і паралельний перенос відповідно. Перетворення G має вигляд [1]:

$$G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/f & 0 & 1 \end{pmatrix} \quad (2.1)$$

Пошук відповідного перетворення H для першого зображення. H відповідає H', якщо $Hl = H'l'$, де l і l' - пара відповідних епіполярних ліній, H^* і $H^{*'}$ - такі перетворення площин, що матриці цих перетворень є союзними до матриць перетворень H і H'. Перетворення шукається мінімізацією суми $\Sigma_i\, d\,(Hu_i, H'u'_i)\; i$ на безлічі точкових відповідностей. При цьому H має вигляд $H = (I + H'p'a^t)\, H'M$ ', де a - деякий вектор і $F = [p] \times M$. Вибирається $H = A$, де [1]:

$$A = \begin{pmatrix} a & b & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2.2)$$

що призводить до мінімізації суми $\sum_i (au_i + bv_i + c + u'_i)^2$.

У разі, коли епіполюс лежить всередині зображення, при перетвореннях H і H' воно буде розбиватися на дві незв'язані області (визначаються останньої координатою точки в проективних координатах), що йдуть на нескінченність, з яких вибираються ті, які містять відповідні точки. Області визначаються з наступного умови: нехай P, P' - матриці проекцій на зображення, $\{xi\}$ - набір точок в тривимірному просторі, $u_i \Leftrightarrow u'_i$ - відповідні точки на зображеннях, які є проекціями $\{xi\}$. Тоді якщо $u_i = (u_{i1}, u_{i2}, u_{i3})$ і $u'_i = (u'_{i1}, u'_{i2}, u'_{i3})$, то при всіх i знак $u_{i3}u'_{i3}$ постійний (рис.2.1) [1].

Рисунок 2.1 - Неректіфіцірованная стереопара і результат її ректифікації у вигляді анагліфа [2]

Чітко видно розбіжність між зображеннями як в орієнтації, так і в положенні. В результаті ректифікації зображення були трансформовані таким чином, що відповідні точки розташувалися уздовж одних і тих же рядків.

2.1.2 Алгоритм Маллон і Вела

В роботі [3] пропонується інший спосіб пошуку відповідного перетворення H'. Вважається, що $H^{-T}F'H = F$, де $F' = (1,0,0)_x$. Таким чином, є така система [3]:

$$\begin{bmatrix} (h_{21}h'_{31} - h_{31}h'_{21})h'_{31} - h'_{21} \\ (h_{21}h'_{32} - h_{31}h'_{22})h'_{32} - h'_{22} \\ (h_{21}h'_{33} - h_{31}h'_{23})h'_{33} - h'_{23} \end{bmatrix} = a \begin{bmatrix} f_{11}\ f_{12}\ f_{13} \\ f_{21}\ f_{22}\ f_{23} \\ f_{31}\ f_{32}\ f_{33} \end{bmatrix} \quad (2.3)$$

Рішення знаходять методом найменших квадратів. При таких перетвореннях зображень епіполярние лінії стають паралельні осі x. Після цього шукають коефіцієнти верхніх рядків H і H'. Розглядається якобіан і його ненульові сингулярні числа σ_1 і σ_2 [3]:

$$\boldsymbol{J}(\boldsymbol{K}, \boldsymbol{p}) = \begin{pmatrix} \frac{d\bar{x}}{dx} & \frac{d\bar{x}}{dx} \\ \frac{d\bar{y}}{dy} & \frac{d\bar{y}}{dy} \end{pmatrix} \quad (2.4)$$

Якщо $\sigma_1 > 1$, то в результаті перетворення будуть створюватися додаткові пікселі, якщо $\sigma_1 < 1$ - відбудеться стиснення зображення. Мінімізується величина, де p_i - набір точок на зображенні [3].

$$f(\hat{a}_{11}, \hat{a}_{12}) = \sum_{i=1}^{n} [(\sigma_1(\boldsymbol{J}(\boldsymbol{K}, \boldsymbol{p_i})) - 1)^2 + (\sigma_1(\boldsymbol{J}(\boldsymbol{K}, \boldsymbol{p_i})) - 1)^2] \quad (2.5)$$

2.2 Методи пошуку пов'язаних точок

2.2.1 Локальний метод пошуку

З метою опису схожості двох пікселів і виявлення сполучених точок використовуються рахунки відповідності. Для їх обчислення визначається функція $\varepsilon(p,q)$. Використовуючи значення інтенсивності, рахунки відповідності можна визначити за допомогою абсолютної різниці інтенсивностей пари пікселів $\varepsilon_{SAD}(p,q)$. Для схожих пікселів значення буде малим, в той час як для розрізняються воно буде велике [4] [5].За умови того, що зображення відкалібровані і ректифікованого, пошук сполученого пікселя обмежений одним рядком. Для знаходження відповідності піксель з рядка лівого зображення порівнюється з кожним пікселем відповідного рядка правого зображення (рис.2.2).

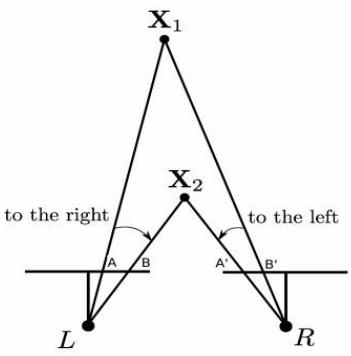


Рисунок 2.2 - Порушення направляючого обмеження для малих об'єктів, розташованих близько до камери [4]

Діапазон пошуку може бути зменшений напрямними обмеженнями [6]: якщо піксель A на лівому зображенні розташований лівіше пікселя B цього ж зображення, то відповідний піксель A_0 на правому зображенні буде так само лівіше відповідного B_0. Однак, ця умова не завжди виконується. Наприклад, воно не застосовується для малих об'єктів, розташованих близько до камери.

Тут об'єкт X_2 розташований правіше X_1, але так як він розташований близько до камери і є точковим об'єктом, то описане умова порушується. Напрямні обмеження дозволяють проводити пошук тільки в заданому діапазоні пікселів і встановити межі значень диспаратности [4]:

$$d \in [d_{min}, d_{max}] \tag{2.6}$$

В результаті пошуку сполученого пікселя може бути отримана представлена кореляційна крива, що показує мінімум рахунків відповідності для заданого пікселя (рис.2.3).

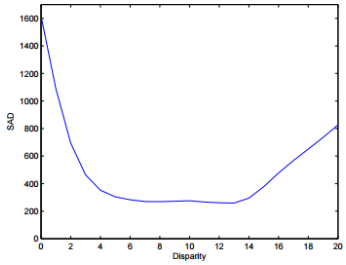


Рисунок 2.3 - Кореляційна крива [4]

Потім, значення диспаратности вибирається методом WTA, без урахування характеру оброблюваної сцени, і в результаті формується карта диспаратности. Розрахунок рахунків вимагає *O(D)* кроків для кожного пікселя. Таким чином, обчислювальна складність локального методу пошуку лінійно зростає з ростом числа пікселів зображення і діапазону диспаратности і виражається як *O(WHD)*, де *W* - ширина зображення, *H* - висота, а *D* - діапазон диспаратности [4].

Алгоритм будує еталонну карту глибину. Еталон стерео відповідності використовується для визначення точності побудови карти глибини. Методика досить проста і зводиться до по піксельному порівнянні значень диспаратности для еталона і отриманої карти глибини і підрахунку числа пікселів, які не задовольняють деякому критерію відповідності. Як правило в якості критерію виступає абсолютна різниця значень і поріг, який ця різниця не може перевищувати. Для зменшення шуму пошук можна проводити з урахуванням сусідніх пікселів. Рахунки відповідності можуть бути підсумовані за допомогою вікна розміром n × n, центр якого розташований в який нас пікселі. Тоді наша метрика набуває вигляду [4]:

$$\varepsilon(p,q) = \sum_{PN \in N_{b,qN} \in N_m} |I_b(p - p_N) - I_m(q - q_N)| \qquad (2.7)$$

де N_b і N_m - вікна для базового і парного зображень відповідно, p_N і q_N - пікселі всередині цих вікон. Подібний підхід передбачає безперервність диспаратности у всьому вікні. Збільшення розмірів вікна веде до зниження шуму на мапі диспаратности, однак різко падає число деталей на зображенні. Крім того, через необхідність враховувати сусідні пікселі, зростає обчислювальна складність алгоритму.

2.2.2 Динамічне програмування для локального методу пошуку

У застосуванні до локальних методів стерео суміщення першим кроком алгоритму є відкидання рядків, які відповідають свідомо неможливим значень d, наприклад, з умов, що піксель на лівому зображенні повинен бути правіше відповідного йому на правому, і з умови $| d | \leq d_{max}$. Застосування динамічного програмування зводиться до пошуку оптимального шляху на отриманої двовимірної матриці. При цьому за кожен тип руху призначається певний штраф. Можливі наступні типи руху: по горизонталі, по вертикалі і по діагоналі. Останні два відповідають затуленим областям, присутнім тільки на одному із зображень стереопари.

Застосування динамічного програмування зводиться до пошуку оптимального шляху на отриманої двовимірної матриці. При цьому за кожен тип руху призначається певний штраф. Можливі наступні типи руху: по горизонталі, по вертикалі і по діагоналі. Останні два відповідають затуленим областям, присутнім тільки на одному із зображень стереопари.

Принцип роботи алгоритму: горизонтальна вісь - вісь *x*, вертикальна - *d*, чорні пікселі відповідають близьким до 0 значенням, скачки *V* і *D* - затуленим областям. Горизонтальні ділянки означають знаходження точної відповідності, а похилі ділянки позначають похилі поверхні (рис.2.4).

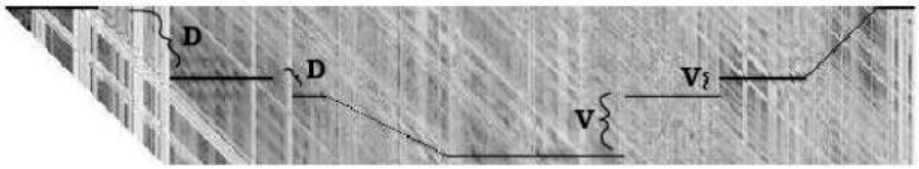


Рисунок 2.4 - Пошук оптимального шляху на отриманої двовимірної матриці [6]

Якщо на зображенні заздалегідь відомі контрольні точки, кількість можливих шляхів різко скорочується за рахунок введення обмеження: шлях повинен проходити так, щоб він не суперечив розстановці контрольних точок. Під контрольними точками, або GCP, розуміють точки, положення і відповідність яких ми можемо визначити досить точно. Можливі також багатозначні GCP - точці одного зображення може відповідати одна точка з невеликого набору варіантів на іншому зображенні [8].

Суттєво підвищити швидкість роботи дозволяє той факт, що кожен рядок обробляється незалежно. Свій внесок вносить і можливість розпаралелювання обчислень. Однак, в результаті функція глибини пікселя може мати гребінчастий вид. Із застосуванням динамічного програмування [6], обчислювальна складність локального алгоритму знижується в порівнянні із загальним методом і не залежить від діапазону диспаратности: *O(WH)* [8].

2.2.3 Динамічне програмування для глобального методу пошуку

В [8] пропонується застосування ідеї динамічного програмування для мінімізації глобального функціоналу:

$$E(f) = E_{data}(f) + \lambda E_{smooth}(f); \quad (2.8)$$

$$E_{data}(f) = \sum_{p \in P} m_f(p); \quad (2.9)$$

де $m_f(x) = min\, \varepsilon_{SAD}(p, q)$, P - безліч пікселів в лівому зображенні, p відповідає q згідно конфігурації f. Нехай $G = (P, N)$ - граф з вершинами-пікселями. Кожен піксель (крім крайніх) з'єднаний з чотирма своїми сусідами (таким чином N - безліч сусідніх пікселів). Тоді [8]:

$$E_{smooth}(f) = \lambda \sum_{(P,PN) \in P} s(d(p), d(p_N)) \quad (2.10)$$

де $s(d_1, d_2)$ - функція гладкості, яка парі значень диспаратности ставить у відповідність штраф за значну відмінність в їх значеннях. Якщо ми якимось чином виберемо дерево-підграф G (с виділеним коренем root) - $G_0 = (P, N_0)$, то алгоритм мінімізації буде виражатися рекуррентной формулою [8]:

$$E_p(d(par(p))) = min_{d(p) \in D}(m_f(p) + s(d(p), d(par(p))) + \sum P_N \in child(p) E_{P_N}(d(p))) \quad (2.11)$$

При цьому мінімум береться по всіх можливих присвоювання пікселуузлу p диспаратности $d(p)$, D - безліч можливих розбіжностей. У разі p = root - корінь, доданок $s(d'(p), d'(par(p)))$ вважається рівним 0. У процесі мінімізації $E_{root}(d'(root))$ для кожного пікселя p вибирається розбіжність $d(p) \in D$, таким чином визначається оптимальна конфігурацію f. Як G_0 автор статті пропонує два варіанти MID Tree і MIDDT Tree. MID Tree - це найменше кістяк G, побудоване з того розрахунку, що вага ребра дорівнює $w_{pN} = |I(x) - I(p_N)|$.

Так як багато ребра матимуть однакові ваги, отримане дерево не буде унікально. Існує інший принцип побудови дерева: для кожної точки визначається функція $D(x)$, яка пікселу зіставляє, наскільки далеко від кордонів він міститься в певної однорідної (наприклад, за інтенсивністю) області. Тоді вага [8]:

$$u_{pp_N} = d_{max} - \frac{1}{2}\big(D(p) + D(p_N)\big), \tag{2.12}$$

$$\text{Де } d_{max} = max_{p \in P} d(p) - \frac{1}{2}\big(D(p) + D(p_N)\big), \tag{2.13}$$

$$\text{Тоді } D(x) = max_{\hat{x} \in P} d(p) - dist(x, \hat{x}). \tag{2.14}$$

Можна вибирати ваги не v_{ppN} або u_{ppN}, а їх комбінацію:

$$C_{pp_N} = kv_{pp_N} + u_{pp_N}, \tag{2.15}$$

$$\text{Де } k = max_{(p,p_N) \in N} + u_{pp_N}. \tag{2.16}$$

Мінімального кістяка дерево, побудоване з урахуванням цих ваг - MIDDT [8]. Згідно [9], обчислювальна складність алгоритму: $O((WH)^3 log(WH))$.

2.2.4 Алгоритми на основі випадкових полів Маркова. Алгоритм розрізу графа

У завданнях стерео зору одним з найпопулярніших способів моделювання є використання випадкового поля Маркова [12]. Як правило, його представляють графом, де вершини, які є випадковими змінними, відповідають пікселям лівого зображення, а ребра графічно зображують залежності між ними. Важливо, що залежності між змінними симетричні: якщо випадковим змінним *X* і *Y* відповідають сусідні вершини графа, то $P(X \mid Y) = P(Y \mid X)$.

Основна властивість випадкових полів Маркова - локальне властивість Маркова - полягає в тому, що кожна змінна залежить тільки від своїх сусідів [8].

Мінімізація функціоналу за допомогою розрізу графа [10], [11] ґрунтується на базових поняттях: *α-expansion* і *αβ-swap*. Для деякого кінцевого безлічі міток L і деякого безлічі пікселів P кожному пікселю $p \in$ P присвоюється певна мітку $l_p \in L$. $P_l = \{p \in P \mid l_p = l\}$, $f = \{Pl \mid l \in L\}$ - поточна конфігурація (в разі пошуку пов'язаних точок мітки можуть бути значеннями диспаратности [10]).

Вершини - пікселі з $P_\alpha \cup P_\beta$, ребра - складові функціоналу: ребра між вершинами пікселями позначають складові, які залежать тільки від двох пікселів. Ребра між виділеними

вершинами і вершинами-пікселями - складові, які залежать від вершини і її мітки. Для α-expansion будується аналогічний граф з виділеними вершинами α і не-α (рис.2.5).

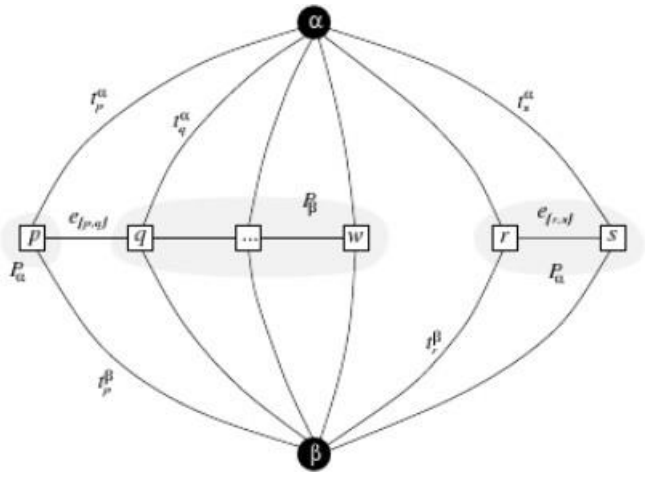


Рисунок 2.5 - Граф для αβ-swap [8]

Нехай дано мітки $\alpha, \beta \in L$. Перехід до конфігурації f_0 називається *αβ-swap*, якщо $P_l = P'_l$ для всіх міток $l \neq \alpha, \beta$. Нехай дана мітка $\alpha \in L$. Перехід до конфігурації f_0 називається α-expansion, якщо $P_\alpha \subset P'_\alpha$ і $Pl \subset P'_l$ для будь-яких $l \neq \alpha$. Відповідно *αβ-swap (f)* *(α-expansion (f))* - множина конфігурацій, в які можна перейти за один *αβ-swap (α-expansion)*. Алгоритм мінімізації функціоналу з використанням *αβ-swap (f)* складається в послідовному зменшенні функціоналу.

1) Задається довільна конфігурація *f*;
2) для кожної пари міток $\{\alpha, \beta\} \subset L$

$$\hat{f} := \arg max_{f' \in \alpha\beta-swap(f)} E(f'); \quad (2.17)$$

3) якщо $E(f') = E(f)$, то присвоюється $f := f$ і здійснюється повернення до кроку 2, інакше - перехід до кроку 4;
4) повертається конфігурація *f*. Алгоритм мінімізації функціоналу з використанням α-expansion аналогічний представленому.

Алгоритм мінімізації функціоналу з використанням *α-expansion* аналогічний представленому. Мінімізація на кожному кроці відбувається за допомогою мінімального розрізу графа (рис2.6.1). Розрізу відповідає нове присвоювання міток α і β (α і не-α - у разі *α-expansion*), тобто нова конфігурація. Алгоритм знаходить локальний мінімум функціоналу.

В роботі [10] відзначається, що алгоритм на основі *α-expansion* показує кращі результати, ніж на основі *αβ-swap* [8]. У гіршому випадку завдання стерео реконструкції з використанням методу розрізу графа може бути вирішена за поліноміальний час $2^{O(log(n))}$ [10].

2.2.5 Алгоритми на основі випадкових полів Маркова. Алгоритм поширення довіри

Алгоритм поширення довіри - стандартний спосіб вирішення задачі максимізації апостеріорної ймовірності в байесовских мережах довіри і MRF за умови відсутності циклів. У моделі, заснованої на випадкових полях Маркова, кожен піксель лівого зображення - це випадкова змінна, і йому зіставляється вузол MRF. Як сусідніх вузлів зазвичай розглядаються чотири сусідніх пікселя. Так як в одержуваної моделі велике кількість циклів, використовується модифікація Loopy Belief Propagation [13], яка відрізняється тим що ігнорується наявність циклів і обробка виконується не за один прохід, а итеративно, до збіжності.

Алгоритм поширює по мережі величини, звані повідомленнями. Наприклад, $m_{pq}(f_p, f_q)$ - повідомлення, що відправляється вершиною X_p вершині X_q - наскільки змінна X_p "думає", що X_q в належному стані.

Складність алгоритму поширення довіри: $O(WHD^2T)$, де T - число ітерацій [14]:

1) всі повідомлення ініціюються однаковим розподілом;
2) повідомлення оновлюються кілька разів [14]:

$$m_{pq}^{i+1}(f_q) \leftarrow kmax_{\varphi pq}(f_p, f_q)\varphi_p^i(f_p)\prod_{f_k \in N(f_p)-q} m_{kp}^i(f_p); \quad (2.18)$$

3) обчислюються свідоцтва 14:

$$b_p(f_p) \leftarrow k_{\varphi_p}(f_p)\varphi_p^i(f_p)\prod_{f_k \in N(f_p)} m_{kp}(f_p); \quad (2.19)$$

4) за свідченнями обчислюється найбільш ймовірна конфігурація

$$f_p^{map} = arg\ max_{b_p}(f_k). \quad (2.20)$$

2.2.6 Полуглобальний алгоритм

Алгоритм полуглобального зіставлення або Semi-Global Matching був запропонований Хіршмюллером в 2005 році в статті [15]. Він заснований на ідеї по піксельного пошуку пов'язаних точок і апроксимації глобальної двовимірної функції гладкості шляхом комбінування безлічі одновимірних функцій [15].

Рахунок відповідності розраховується для пікселя p базового зображення на підставі його інтенсивності I_{bp} і передбачуваного відповідності IMG для пікселя q парного зображення.

Один із способів по піксельного підрахунку рахунків полягає в методі, запропонованому Берчфілд і Томасі [16]. Рахунок C_{BT} (p,d) розраховується як абсолютна мінімальна різниця інтенсивностей в точках p і q в діапазоні полпіксела в кожному напрямку вздовж епіполярной лінії. Альтернативою є розрахунок рахунків відповідності, заснований на взаємній інформації *MI* [14]. Вона визначається з ентропії H двох зображень і їх взаємної ентропії [16]:

$$MI_{i1,i2} = H_{I_1} + H_{I_2} - H_{I_1,i_2} \tag{2.21}$$

Ентропії розраховуються з розподілу інтенсивностей P оброблюваних зображень 16:

$$H_I = -\int_0^1 P_I(i) l \log P_I\,(i) di \tag{2.22}$$

$$H_I = -\int_0^1 P_{I1,I2}(i_1, i_2) \log P_{I1,I2}(i_1, i_2) di_1\, di_2 \tag{2.23}$$

Для завдань стерео зіставлення одне із зображень має бути перетворено у відповідності з картою диспаратности для досягнення відповідності другому зображенню, щоб пов'язані пікселі на кожному зображенні перебували в одній і тій же позиції, тобто $I_1 = I_b$ і $I_2 = f_D$ (I_m). З метою вирішення цієї проблеми було запропоновано ітеративний підхід, в якому перша ітерація починається з формування довільної карти диспаратности для розрахунку CMI. Отримані рахунки потім використовуються для зіставлення обох зображень і розрахунку нової карти диспаратности, яка служить основою для наступної ітерації. Для якісної роботи алгоритму досить

трьох ітерацій, оскільки навіть помилкова карта диспаратности дозволяє зробити хорошу оцінку розподілу *P*.

Крім того, розрахунок починається для зображень, зменшених до 1/16 оригінального розміру. На кожній наступній ітерації карта глибини масштабується і використовується для розрахунків. Якщо загальна складність алгоритму *O(WHD)*, то час його виконання для половинного дозволу знижується в 23 = 8 разів [15]. Для трьох ітерацій при вирішенні 1/16 і подальшому його збільшенні розрахунковий час збільшується незначно [15]:

$$1 + \frac{1}{2^3} + \frac{1}{4^3} + \frac{1}{8^3} + 3\frac{1}{16^3} \approx 1{,}14, \quad (2.24)$$

За піксельний розрахунок рахунків в основі своїй неоднозначний, і невірні відповідності можуть мати меншу вагу, ніж вірні, зважаючи на наявність шуму, рівномірних ділянок, відблисків і т.д. У зв'язку з цим вводиться додаткове обмеження, яке забезпечує гладкість, призначаючи штраф для зміни диспаратности в сусідніх пікселях [15]:

$$E(d) = \sum_p C(\boldsymbol{p}, d(p)) + \sum_{p_N \in N_p} P_1 \boldsymbol{T}[|d(p) - d(p_N| = 1] + \sum_{p_N \in N_p} P_2 \boldsymbol{T}[|d(p) - d(p_N| > 1], \quad (2.25)$$

де *T []* - оператор, який дорівнює 1 якщо його аргументом є «істина», і 0, якщо «брехня». Перший член - сума всіх рахунків для диспаратность *d*. Другий член накладає штраф P_1 для всіх пікселів *q* в околиці *Np* пікселя *p*, для яких диспаратность трохи змінюється. Третій член накладає більший штраф P_2 на все більші зміни диспаратности. Використання меншого штрафу для малих змін забезпечує адаптацію до похилих і кривих поверхонь. Штраф для всіх великих змін забезпечує захист від розривів [17]. Розриви часто видно як зміни інтенсивності.

2.3 Method Shape from focus

Метод shape-from-focus використовує різні рівні фокусування для отримання послідовності зображень об'єкта [18].

Всі світлові промені, які випромінюються точкою об'єкта *P* і перехоплені лінзою, переломлюються лінзою, щоб сходитися в точці *Q* на площині зображення. Співвідношення між відстанню до об'єкта *o*, фокусною відстанню об'єктива *f*, і відстанню зображення *i*, визначається законом Гаусса [18]:

$$\frac{1}{o} + \frac{1}{i} = \frac{1}{f}. \tag{2.26}$$

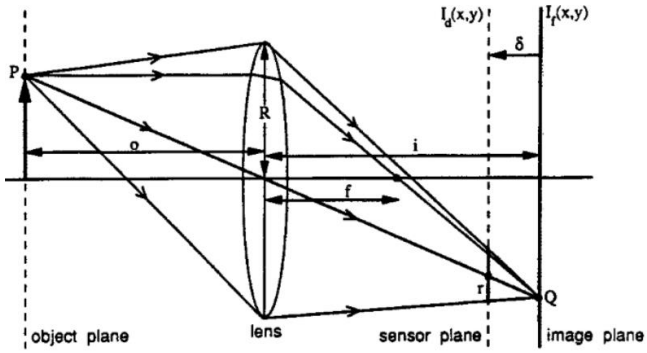


Рисунок 2.6 - Формування фокусированного і расфокусированного зображення [18]

Кожна точка на площині об'єкта проектується на одну точку на площині зображення, що призводить до формування чіткого або сфокусованого зображення $I_f(x,y)$ на площині зображення. Однак якщо площину датчика не збігається з площиною зображення і зміщена від неї на відстань *σ*, енергія отримана від об'єкта лінзою, розподіляється по круглому плямі на площині датчика. Установка взаємозв'язку між радіусом *r* круглого плями і зміщенням датчика *σ* [6]:

$$r = \frac{\delta R}{i} \tag{2.27}$$

Де *R* - радіус лінзи. Також можна визначити, що радіус *r* круглої плями не залежить від положення *P* на площині об'єкта. Розподіл енергії світла по круговому плями або функція розмиття можна моделювати за допомогою фізичної оптики [6]. Дуже часто двовимірна функція Гаусса використовується для апроксимації фізичної моделі [6] [9]. Потім розмите або розфокусовані зображення $I_d\ (x,\ y)$, сформований на площині датчика, можна описати як результат згортання сфокусованого зображення *(x,y)* з функцією розмиття *h (x,y)* [6]:

$$I_d(x,y) = h(x,y) * I_f(x,y) \tag{2.28}$$

$$\text{Де } h(x,y) = \frac{1}{2\pi\delta h^2} e^{-\frac{x^2+y^2}{2\delta h^2}} \tag{2.29}$$

де σ_h, параметр розкиду, передбачається пропорційним радіусом r. Константа пропорційності залежить від оптики, вибірки і так далі. Дефокусування спостерігається як для позитивних, так і для негативних зсувів датчика. Проаналізувати процес расфокусировки в частотної області (u,v). Якщо $I_F(u,v)$, $H(u,v)$ і $I_D(u,v)$ є перетвореннями Фур'є $I_f(x, y)$, $h(x, y)$ і $I_d(x, y)$ відповідно, ми можемо висловити як [18]:

$$I_D(u,v) = H(u,v)\, I_F(u,v) \tag{2.30}$$

$$\text{Де } H(u,v) = -\frac{u^2+v^2}{2}\delta h^2 \tag{2.31}$$

$H(u,v)$ пропускає низькі частоти, в той же час послаблюючи високі частоти в фокусованому зображенні. Крім того, у міру того, як зсув σ датчика збільшується, радіус расфокусировки r збільшується, а параметр розсіювання σ_h збільшується. Отже, расфокусировка це процес фільтрації нижніх частот, коли смуга пропускання зменшується зі збільшенням расфокусировки [18].

Так розфокусовані зображення об'єкта може бути отримано трьома способами: шляхом зміщення датчика відносно площини зображення, шляхом переміщення об'єктива або шляхом переміщення об'єкта відносно площини об'єкта [18].

2.4 Пошук ключових точок

2.4.1 Алгоритм Speeded-Up Robust Features (SURF)

SURF вирішує завдання пошуку особливих точок і створення їх дескрипторів, інваріантних до масштабу і обертанню - при зміні масштабу або поворотах зображення, описи ключових точок не змінюються. Метод здійснює пошук особливих точок за допомогою матриці Гессе. Її визначник досягає екстремуму в точках максимального зміни градієнта яскравості.

Виходячи з цього, отримуємо, що такі елементи зображень як плями, кути і краю ліній будуть досить добре «впізнавані» [19] [20].

$$Gs(\big(f(x,y)\big) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}, \det(Gs) = \frac{\partial^2 f \partial^2 f}{\partial x^2 \partial y^2} - (\frac{\partial^2 f}{\partial x \partial y})^2 \quad (2.32)$$

Де за формолою (2.32) *Gs* - матриця Гессе, *f (x,y)* - функція зміни градієнта яскравості. Инвариантность масштабу SURF забезпечується використанням різномасштабних фільтрів при знаходженні гессіанов. Для кожної особливої точки розраховується масштаб і напрямок максимального зміни яскравості, формується дескриптор, що представляє собою набір з 64 (або 128) чисел. Для забезпечення ефективності обчислень кожне зображення, перед «запуском» алгоритму, представляється в інтегральному форматі. Даний формат передбачає використання матриці, розмірність якої еквівалентна розміру самого зображення.

$$I(x,y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(i,j) \quad (2.33)$$

Де за формолою (2.33) *I (i,j)* - яскравість пікселя вхідного зображення, *(x,y)* координати пікселя на матрицю. Алгоритм SURF більш ефективний при аналізі зображень з розмиванням і наявністю обертань. Обробка ж зображень з сильною зміною освітленості створить певні труднощі для даного алгоритму. Також метод не «спрацює» при обробці простих (з однотонною текстурою) об'єктів. SURF сприймає картинку як єдине ціле і в таких умовах здійснює пошук його особливостей. Точки будуть виявлені на кордоні об'єкта з фоном і при зміні об'єкта виникнуть проблеми з його ідентифікацією [19].

2.4.2 Алгоритм Oriented FAST and Rotated BRIEF (ORB)

Алгоритм ORB опублікований в 2011р [20] [21]. В його основі лежить комбінація таких алгоритмів, як детектор FAST (Features from Accelerated Segment Test) і дескриптор BRIEF (Binary Robust Independent Elementary Features) з деякими поліпшеннями.

Опис алгоритму (FAST) виглядає наступним чином [23].

1) Позначення пікселя на зображенні. Припустимо, інтенсивність цього пікселя *IP*. Це піксель, який слід ідентифікувати як цікавий пункт чи ні (рис.2.7).

2) Встановлення значення порогової інтенсивності *T*.

3) Коло з 16 пікселів, що оточують піксельну *p*.

4) "*N*" суміжних пікселів з 16 повинні бути над або нижче *IP* за значенням *T*, якщо піксель необхідно визначити як точку інтересу.

5) Щоб алгоритм був швидким, спочатку порівнюються інтенсивність пікселів 1, 5, 9 та 13 кола з *IP*. Як видно з (рис.2.7), щонайменше три з цих чотирьох пікселів повинні задовольнити значення пороговий критерій, щоб точка інтересу існувала.

6) Якщо принаймні три з чотирьох піксельних значень - *I1, I5, I9 I13* не вище або нижче *IP + T, P* не цікава точка. У цьому випадку відхиліть піксель *p* як можливу цікаву точку. Інакше, якби щонайменше три пікселі є вище або нижче *Ip + T*, потім перевірте всі 16 пікселів і перевірте, чи немає у критерій потрапляє 12 суміжних пікселів.

7) Повторіть процедуру для всіх пікселів на зображенні.

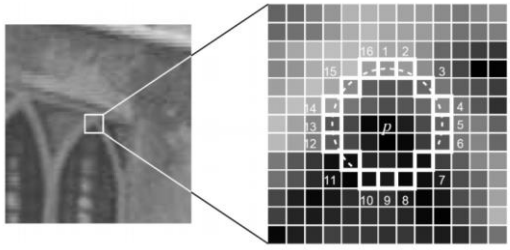


Рисунок 2.7 - Зображення, яке показує тестову точку інтересу та 16 пікселів на колі.

Для знаходження особливих точок використовується детектор FAST, який неодноразово демонстрував себе в якості швидкого інструменту при виявленні особливостей зображення. Метод FAST не інваріантний до поворотів, а тому, виникає необхідність введення параметра кутовий орієнтації. Зазначений параметр забезпечує детектування при обертанні об'єкта, використовує напрямок з найбільшою інтенсивністю, яка призначається орієнтацією особливої точки θ.

Опис методу (BRIEF) виглядає наступним чином [22]:

Визначимо тест τ на позицій p розміру $S \times S$ як [22]:

$$\tau(p; x, y) \coloneqq \begin{Bmatrix} 1 & if\ p(x) < p(y) \\ 0 & otherwise \end{Bmatrix}, \tag{2.34}$$

де $p(x)$ - інтенсивність пікселів в згладженої версії p при $x = (u, v)^T$. Вибір набору $n_d(x, y)$ позиційні пар однозначно визначає набір бінарних тестів. Береться BRIFF дескриптор як ланцюжок бітів n_d-вимірювання [22]:

$$f_{n_d} := \sum_{1 \le i \le n_d} 2^{i-1} \tau(p; x_i, y_i) \quad (2.35)$$

Дескриптор BRIEF, що представляє собою вектор довжиною, рівній 256, складається з результатів бінарних тестів навколо особливої точки. Для досягнення інваріантності до обертання область обчислення дескриптора визначається орієнтацією особливої точки θ. Дескриптор BRIEF забезпечує розпізнавання об'єкта з різних ракурсів при досить малих витратах на обчислення [22].

Швидкість роботи алгоритму ORB вище, ніж у алгоритму SURF, так як використовуються в повному обсязі особливі точки, а тільки та їх частина, яка залишається після «відсіву» детектора кутів Харріса.

2.5 Алгоритм реконструкцій із хмари точок

Для кожного пікселя в зображенні розраховуються координати в тривимірному просторі і призначаються кольори з RGB. Виконання перетворення вхідних параметрів в хмару точок об'єкта виглядає наступним чином [24]:

1) Переклад матриці параметрів камери в матрицю відповідностей, для подальших перетворень значень з карти глибин;
2) Поточечное приведення точок з карти глибин використовуючи параметри положення в просторі і масштабних характеристик об'єкта;
3) Занесення точок об'єкта в тривимірну проекцію дотримуючись положення в просторі;
4) Перенесення пікселів кольорового зображення лівої частини стереозображення на відповідні точки в проекції;
5) Перенесення хмари точок на віртуальну сцену [24].

Перенесення точок здійснюється за формулою [24]:

$$sm' = A[R|t]M' \tag{2.36}$$

або в наведеному вигляді [24]:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r11 & r12 & r13 & t1 \\ r21 & r22 & r23 & t2 \\ r31 & r32 & r33 & t3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.37}$$

де:

1) (X, Y, Z) - координати 3D точки в глобальному просторі;

2) (u, v) - координати проектування точок на пікселі;

3) A - це камера матриці, набір її параметрів;

4) (cx, cy) - точка, прийнята згідно з центральною для зображення;

5) fx, fy - фокусні відстані певні для поточної точки.

2.6. Оцінка і порівняння існуючих методів.

В результаті проведеного аналізу було встановлено, що найбільший вплив на обчислювальну складність і точність побудови об'ємної моделі надає етап пошуку відповідність точок. Саме в цій області ведеться більшість розробок, метою яких є оптимізація тих чи інших характеристик методів тривимірної реконструкції.

Проведемо порівняльний аналіз. Найбільшу точність дають методи стерео зіставлення. Вони ж має найбільшу обчислювальну складність. Де алгоритми локального пошуку істотно поступається в точності побудови об'ємної моделі. Незважаючи на те, що своє визнання він заслужив багато в чому завдяки швидкості своєї роботи, ряд алгоритмів на даний момент практично не поступається йому, забезпечуючи при цьому куди менший відсоток помилок. До таких алгоритмам в першу чергу відносяться метод динамічного програмування (заснований на локальному методі) і полуглобальний алгоритм.

Істотний вплив на результат можуть зробити методи обробки карти глибини. Їх застосування не завжди виправдане, тому що вони часто збільшують обчислювальну складність

всього алгоритму, але при використанні методів пошуку з порівняно невисокою обчислювальною складністю, можна домогтися значних результатів у підвищенні якості формованої моделі.

Одним і популярних є методи ректифікації. Методи ректифікації в більшості своїй мають приблизно однакову швидкість роботи і дають схожі результати. Застосування більш складних методів абсолютно невиправдано, тому що основна мета ректифікації спростити роботу на етапі пошуку пов'язаних точок.

Метод форми від фокусу є більш вузько спрямовані і використовують різні рівні фокусування для отримання послідовності зображень об'єкта.

Алгоритми побудова тривимірної моделі з хмари точок є простими і мають невисоку обчислювальну складність. І мають широке застосування в тривимірній реконструкцій.

Нижче наводиться таблиця 2.1, в якій відображені ключові характеристики досліджених методів.

Таблиця 2.1 - Характеристики досліджених методів

Назва методу	Обчислювальна складність	Точність
Локальний метод пошуку.	O(WHD)	73%
Динамічне програмування для локального методу пошуку.	O(WH)	85%
Динамічне програмування для глобального методу пошуку.	$O((WH)^3 log(WH))$	86%
Алгоритми на основі випадкових полів Маркова. Алгоритм розрізу графа.	$2^{(log(n))}$	92%
Алгоритми на основі випадкових полів Маркова. Алгоритм поширення довіри.	$O(WHD^2T)$	79%
Полуглобальний алгоритм	O(WHD)	88%

2.7 Оцінка комбінування методів

Для досягнення поставленої задачі, методи та алгоритми можуть бути скомбіновані для підвищення точність побудови тривимірної моделі. Такими прикладом комбінування є алгоритми:

1) Oriented FAST and Rotated BRIEF (ORB). В його основі лежить комбінація таких алгоритмів як детектор FAST (Features from Accelerated Segment Test) і дескриптор BRIEF (Binary Robust Independent Elementary Features). Так в цьому алгоритмі було запропоновано спочатку обчислювати орієнтацію особливої точки і потім проводити бінарні порівняння вже у відповідність з цією орієнтацією [20].

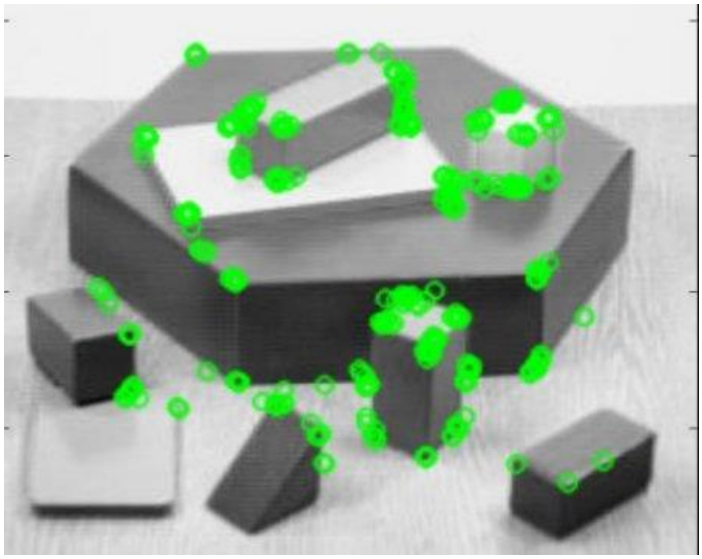


Рисунок 2.8 - Приклад реконструкцій алгоритму Oriented FAST and Rotated BRIEF [20]

2) Гібридний паралельний алгоритм зіставлення стереозображення. Так на стереозображення знаходяться на так званих епіполярних лініях, які можуть бути визначені з використанням заданої або обчисленої фундаментальної матриці. На другому етапі визначення відповідних точок передує етап ректифікації зображень, в результаті якого епіполярние лінії перетворюються в паралельні прямі [25].

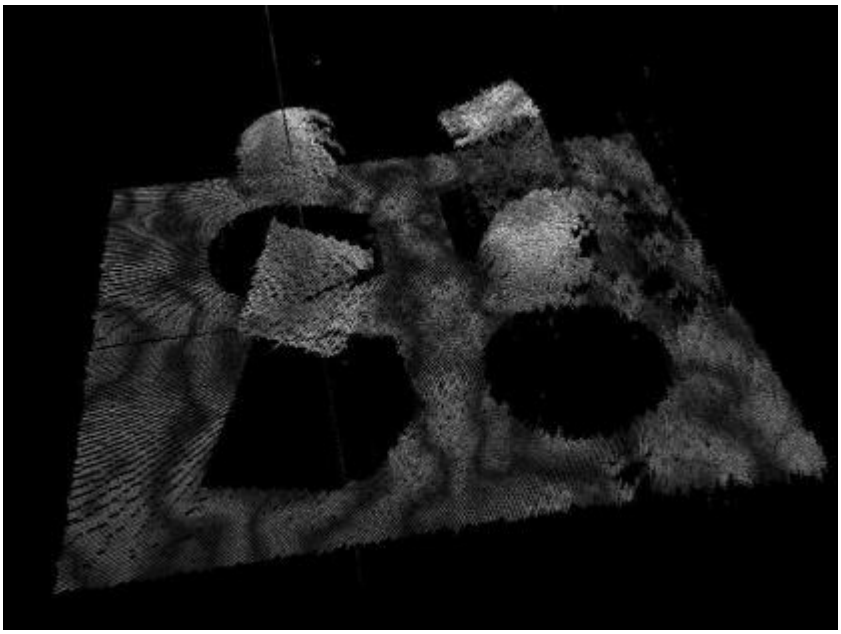


Рисунок 2.9 - Приклад гібридного паралельного алгоритму зіставлення стереозображення [25]

3) Локальний метод пошуку. Першим етапом буде знаходження відповідності піксель з рядка лівого зображення порівнюється з кожним пікселем відповідного рядка правого зображення. Потім, значення диспаратности вибирається методом WTA, без урахування характеру оброблюваної сцени, і в результаті формується карта диспаратности [15].

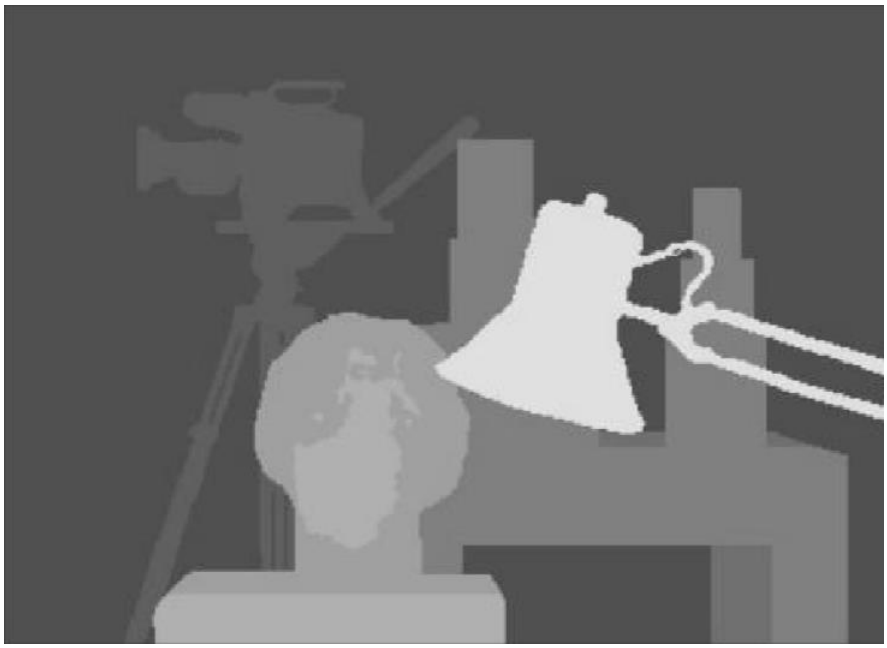


Рисунок 2.10 - Приклад локального методу пошуку [15]

Комбіновані алгоритми показують хороший результат. Але виходячи з аналізу можна зробити висновок що комбінування може призвести до зростання обчислювальної складності, але відбуватися збільшення точності реконструкцій моделі.

Так було прийнято рішення щоб підвищити точність побудови моделі, потрібно комбінування методів та алгоритмів. Для досягнення поставленої задачі був обраний метод стерео зіставлення, і алгоритм побудова тривимірної моделі з хмари точок. Метод стерео зіставлення має більш високу обчислювальну складність, але дають більш високу точність побудова тривимірної моделі. У свою чергу алгоритм побудова моделі з хмари точок має не велику обчислювальну складність, недоліком алгоритму є можливість погіршення точності об'єкт внаслідок втрати знаходження вершин точок на кордоні об'єкта. Таким чином за допомогою методу стерео зіставлення знаходимо відповідність між точками двох зображень. Якщо збігаються 3 кольору точок (RGB) на обох зображеннях то формується вершини точок, на виході отримуємо хмара точок об'єкта.

2.8 Висновки до другого розділу

У другому розділі було проведено аналіз сучасних алгоритмів і методів тривимірної реконструкцій. Виходячи з аналізу можна зробити висновок що тривимірна реконструкція має безліч методів і алгоритмів. Кожен алгоритм і метод являє собою набір етапів обробки зображення. І в залежності поставлених завдань використовується той чи інший метод. Була проведена оцінка і порівняння методів. Аналіз показав що високу точність дають методи стерео зіставлення. Також була проведена оцінка комбінування методів. З оцінки можна зробити висновок що, комбінування дозволяє досягти більш високих результатів. Так прикладом комбінування є Алгоритм Oriented FAST and Rotated BRIEF (ORB). Так для досягнення поставленої задачі, було прийнято рішення про комбінуванні методу стерео зіставлення і алгоритму побудова тривимірної моделі з хмари точок.

2.9 Перелік джерел посилань до розділу 2

1) Hartley, R.I. Theory and Practice of Projective Rectification. / R.I. Hartley.

2) Uncalibrated Stereo Image Rectification. Посилання: https://www.mathworks.com/help/vision/examples/uncalibrated-stereo-image-rectification.html.

3) Mallon, J. Projective Rectification from the Fundamental Matrix. / J. Mal- lon, P. Whelan // Image and Vision Computing. - 2005. - Vol. 23. - Issue 7. - P. 643-650.

4) Ndhlovu T. An investigation into stereo algorithms: An emphasis on local- matching. / University of Cape Town, 2011, P 75.

5) Hirschmuller, H., Scharstein, D. Evaluation of stereo matching costs on images with radiometric differences. / IEEE Transactions on Pattern Anal- ysis and Machine Intelligence. - 2009. - Vol. 31. - № 9. - P. 1582–1599. 26.

6) A. L.Yuille, T. Poggio, “A generalized ordering constraint for stereo correspondence,” MIT, A.I.Memo 777, 1984, aI Lab. 7, P 27.

7) Бінокулярне зрение і стереопсис. Посилання: https://habr.com/ru/post/30282/.

8) Вахітов А.Т., Гуревич Л.С., Павленко Д.В. Обзор алгоритмів стереоречення. Посилання: http://www.math.spbu.ru/user/gran/soi4/pavlenko.pdf.

9) Tomasi C. Global Stereo in Polynomial Time. Посилання: https://users.cs.duke.edu/~tomasi/papers/tomasi/tomasiYork07.pdf.

10) Kolmogorov V., Zabih R. Computing visual correspondence with occlu- sions using graph cuts // ICCV. Vol. 2. 2001. PP. 508-515.

11) Boykov Y., Veksler O., Zabih R. Fast approximate energy minimization via graph cuts // IEEE TPAMI 23 (11). 2001. PP. 1222-1239.

12) Rudnick A. Stereo Vision and Markov Random Fields. Посилання: https://www.cs.indiana.edu/~alexr/nonpubs/alexr-mrf.pdf.

13) Sun J., Shum H., Zheng .N Stereo matching using belief propagation // In ECCV. 2002. PP. 510–524.

14) E. Riegler, G. E. Kirkelund, C. N. Manch´on, Badiu, et al. Merging Belief Propagation and the Mean Field Approximation: A Free Energy Approach. Посилання: https://arxiv.org/pdf/1112.0467.pdf.

15) H.Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. IEEE Conference on Computer Vision and Pattern Recognition, 2006.

16) S. Birchfield, C. Tomasi. Depth discontinuities by pixel to pixel stereo. In Proceedings of the Sixth IEEE International Conference on Computer Vi- sion, pages 1073–1080, Mumbai, India, January 1998.

17) Y. Boykov, O. Veksler, and R. Zabih. Efficient approximate energy mini- mization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (11): 1222- 1239, 2001.

18) Shree K. Nayar. Shape from Focus System. Посилання: http://graphics.stanford.edu/courses/cs348b-06/homework3/Nayar_CVPR92.pdf.

19) Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. Посилання: https://www.vision.ee.ethz.ch/~surf/eccv06.pdf.

20) Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski ORB: anefficient alternative to SIFT or SURF. Willow Garage, Menlo Park, California, 2011. 8 p.

21) E. Karami, S. Prasad, M. Shehata Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images. Faculty of Engineering and Applied Sciences, Memorial University, Canada, 2015. 10 p.

22) Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary Robust Independent Elementary Features.

23) Deepak Geetha Viswanathan. Features from Accelerated Segment Test (FAST).

24) Юркін В.А. Системи побудови 3D моделі особи людини з двох зображень зі стереопари і подальшим порівнянням з еталоном

25) Фурсов В.А., Гошин Є.В., Котов О.П. Гібридний паралельний алгоритм зіставлення стереозображення

РОЗДІЛ 3

РОЗРОБКА АЛГОРИТМУ ТРИВИМІРНОЇ РЕКОНСТРУКЦІЙ ІЗ ХМАРИ ТОЧОК НА ОСНОВІ МЕТОДУ СТЕРЕО ЗІСТАВЛЕННЯ

У третьому розділі пропонується алгоритм тривимірної реконструкцій із хмари точок на основі зіставлення двох зображень. На вхід алгоритму обробки подається набір з декількох зображень (два або більше), результатом роботи якого є тривимірний об'єкт. Пропонований алгоритм реалізований на мові C ++ з використанням графічної бібліотекою OpenGL. А також розроблена програма яка генерує прості геометричні об'єкти для проведення експериментів.

3.1 Постановка завдання і рішення

Суть математичного рішення задачі відновлення тривимірного об'єкту за двома або більше зображень полягає в описі зміни положення датчиків (камери) захоплення проекції при виконанні різних кадрів. Положення визначається переносом і обертанням. Камери укомплектовані датчиками орієнтації і необхідним програмним забезпеченням, що визначає кути з високою точністю. Перенесення поки досить точно визначити не можна. Тому, завдання тривимірної реконструкції ставиться в рамках, коли становище камери визначається тільки орієнтацією датчика.

Осі системи координат (СК) пристрої виходять з центру пристрій (камеру). Обертання визначається кутами Ейлера які визначають три повороти системи, які дозволяють привести будь-яке положення системи до поточного. Позначимо початкову систему координат як (x, y, z) кінцеву як (*X, Y, Z*) Перетин координатних площин *xy* і *XY* називається лінією вузлів N. Кут *alpha* між віссю x і лінією вузлів - кут прецесії. Кут *beta* між осями z і Z - кут нутації. Кут *gamma* між віссю X і лінією вузлів - кут власного обертання.

Повороти системи на ці кути називаються прецесія, нутація і поворот на власний кут (обертання). Такі повороти некомутативними і кінцеве положення системи залежить від порядку, в якому відбуваються повороти. У разі кутів Ейлера проводиться спочатку поворот на кут *alpha* навколо осі z, потім поворот на кут beta навколо осі N, і останнім поворот на кут gamma навколо осі Z [1].

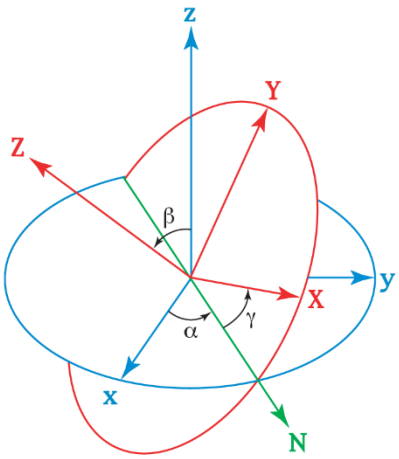
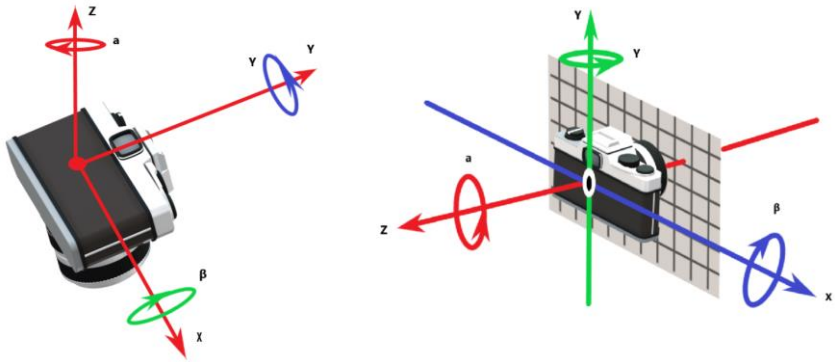


Рисунок 3.1 - Кути Ейлера [1]

Які представляють собою різницю в градусах між координатами пристрою і земними координатами (рис.3.2).



3.2 - Орієнтація пристрою щодо координат

Кут *alpha* = 0 °, коли вісь Y пристрої спрямована на північ. Кути *beta* = 0 ° і *gamma* = 0 °, коли площину пристрою паралельна земній поверхні. При повороті пристрою навколо відповідної осі (рис.3.2) значення кутів *alpha* і *gamma* збільшується в напрямку проти годинникової стрілки, *bema* - за годинниковою стрілкою.

На (рис.3.3) представлена схема для визначення математичної залежності між координатами точок 2-х зображень. Точки описані в площині *XY* (*xy*) головного та допоміжного зображень, вісь *Z* (*z*) спрямована перпендикулярно до площини камери і проходить через її центр. Головним є те зображення, фото якого отримано першим. Значення кутів визначається різницею (дельта) кутів допоміжного положення камери щодо головного:

$$\begin{aligned} \Delta\, a &= a_1 - a_0 \\ \Delta\, \beta &= \beta_1 - \beta_0 \\ \Delta\, Y &= Y_1 - Y_0 \end{aligned} \tag{3.1}$$

Відповідно до рівняннями знак значення для різниці кутів буде тим же, що і для кутів, які визначають положення пристрою відносно земної поверхні (рис.3.3). У подальшому викладі різниця кутів позначається без дельта.

3.2 Визначення відповідності між координатами точок зображень

Положення допоміжної СК *xyz* щодо головної *XYZ* задається композицією 3-х матричних перетворень - обертання щодо осі *Z*, обертання щодо осі *X* і обертання щодо осі *Y*.

У тривимірному просторі перехід з однієї СК до іншої описується в загальному випадку наступним чином [2]:

$$\begin{aligned} x^* &= a_1 x + a_2 y + a_3 z + \lambda \\ y^* &= \beta_1 x + \beta_2 y + \beta_3 z + \mu \\ z^* &= Y_1 x + Y_2 y + Y_3 z + V \end{aligned} \tag{3.2}$$

У матричному поданні рівняння можуть бути записані 2-а способами [2]:

$$(x^* y^* z^* 1) = (xyz1)\begin{pmatrix} a_1 & \beta_1 & Y_1 & 0 \\ a_2 & \beta_2 & Y_2 & 0 \\ a_3 & \beta_3 & Y_3 & 0 \\ \lambda & \mu & \nu & 1 \end{pmatrix}\begin{pmatrix} x^* \\ y^* \\ z^* \\ 1 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_3 & \lambda \\ \beta_1 & \beta_2 & \beta_3 & \mu \\ Y_1 & Y_2 & Y_3 & \nu \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.3}$$

$$R_z R_x R_y$$

$$R_z = \begin{pmatrix} \cos a & \sin a & 0 & 0 \\ -\sin a & \cos a & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \beta & -\sin \beta & 0 \\ 0 & \sin \beta & \cos \beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.4)$$

$$R_y = \begin{pmatrix} \cos Y & 0 & -\sin Y & 0 \\ 0 & 1 & 0 & 0 \\ \sin Y & 0 & \cos Y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матриці записані так, що позитивне значення кутів *alpha* і *gamma* - в напрямку проти годинникової стрілки, *bema* - за годинниковою стрілкою.

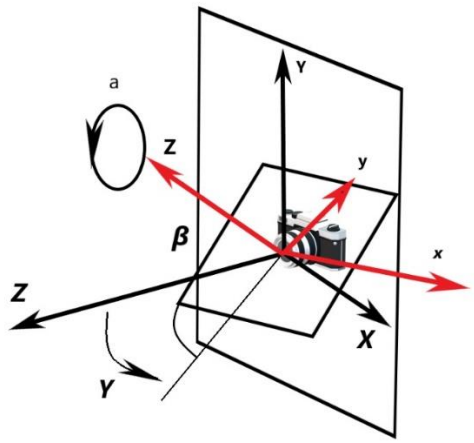


Рисунок 3.3 - Схема взаємозв'язків орієнтації двовимірного зображення

Для перерахунку точок об'єкта з СК *XYZ* в СК *xyz* необхідно отримати результат перемноження зворотних матриць в послідовності навпаки. Зворотні матриці в цьому випадку є матриці, що забезпечують обертання з негативним кутом:

$$R_z^{-1} R_x^{-1} R_y^{-1}$$

$$R_y^{-1} = \begin{pmatrix} \cos Y & 0 & \sin Y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin Y & 0 & \cos Y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} R_x^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \beta & \sin \beta & 0 \\ 0 & -\sin \beta & \cos \beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.5)$$

$$R_y^{-1} = \begin{pmatrix} \cos Y & -\sin Y \sin\beta & \sin Y \cos\beta & 0 \\ 0 & \cos\beta & \sin\beta & 0 \\ -\sin Y & -\cos Y \sin\beta & \cos Y \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_y^{-1}R_x^{-1}R_z^{-1} = \begin{pmatrix} \cos Y \cos a - \sin Y \sin\beta \sin a & -\cos Y \sin a - \sin Y \sin\beta \cos a & \sin Y \cos\beta & 0 \\ \cos\beta \sin a & \cos\beta \cos a & \sin\beta & 0 \\ -\sin Y \cos a - \cos Y \sin\beta \sin a & \sin Y \sin a - \cos Y \sin\beta \cos a & \cos Y \sin Y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.6)$$

Результат множення трьох зворотних матриць і матриці ортогонального перетворення дає наступний результат:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_p^{-1} = \begin{pmatrix} \cos Y \cos a - \sin Y \sin\beta \sin a & -\cos Y \sin a - \sin Y \sin\beta \cos a & 0 & 0 \\ \cos\beta \sin a & \cos\beta \cos a & 0 & 0 \\ -\sin Y \cos a - \cos Y \sin\beta \sin a & \sin Y \sin a - \cos Y \sin\beta \cos a & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.7)$$

Перетворення координат точок з головною СК у допоміжну СК в матричному поданні має вигляд:

$$(xyz1) = (XYZ1)T_p^{-1} \quad (3.8)$$

У підсумку, рівняння перетворення точок просторового об'єкта в точки поточного зображення запишуться у вигляді:

$$x = X(\cos Y \cos a - \sin Y \sin\beta \sin a) + Y \cos\beta \sin a - Z(\sin Y \cos a + \cos Y \sin\beta \sin a)$$

$$y = -X(\cos Y \sin a + \sin Y \sin\beta \cos a) + Y \cos\beta \cos a + Z(\sin Y \sin a - \cos Y \sin\beta \sin a) \quad (3.9)$$

$$Z = 0$$

3.3 Опис кроків роботи алгоритму

Система рівнянь (3.9) дозволяє знайти координату *Z* кожної точки *(X, Y)* на головному зображенні, якщо відомі кути і відповідне місце на *(x, y)* на допоміжному зображенні. Відповідність між точками може встановлюватися за їх кольором. Опис алгоритму:

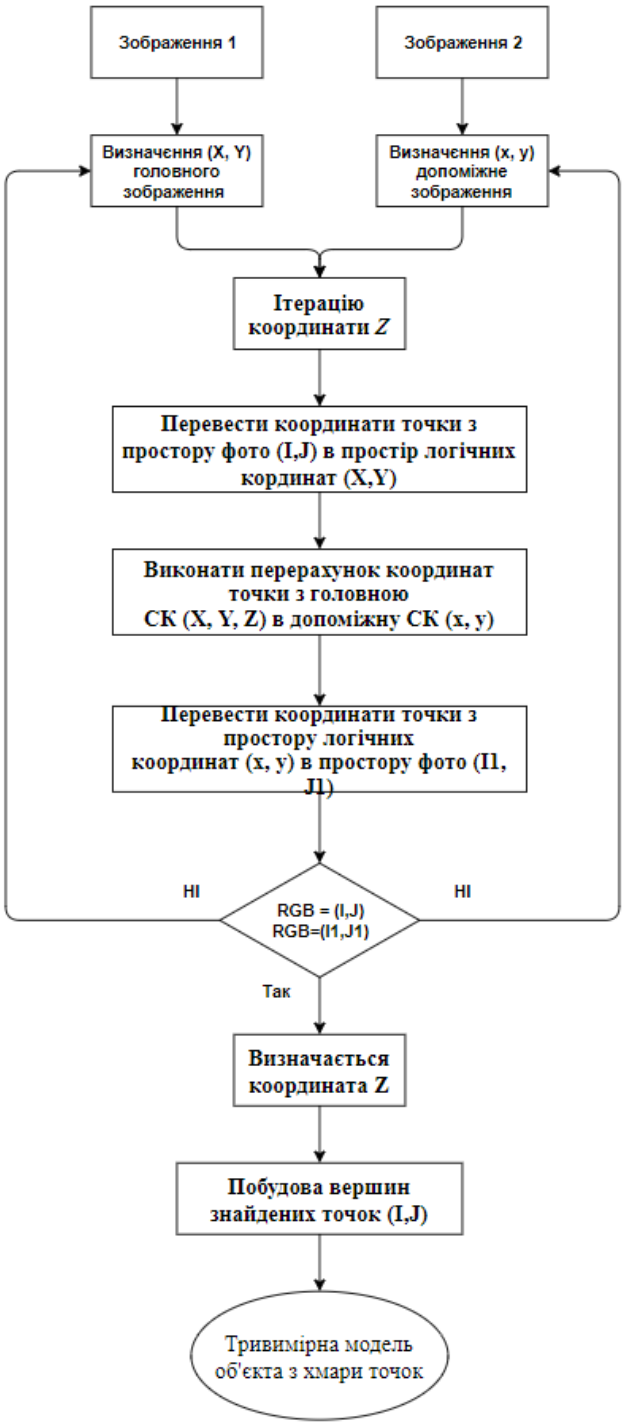


Рисунок 3.4 - Схема алгоритму

1) Для кожної точки головного зображення *(X, Y)* визначаємо відповідну точку на допоміжному зображенні *(x, y)*, здійснюючи ітерацію координати *Z*.
2) Відповідна точка знайдена, якщо збігаються 3 кольору точок (RGB) на обох зображеннях.
3) Разом з відповідною точкою визначається і координата *Z*.

Для пошуку потрібної точки на двох зображеннях необхідно в ітераційному режимі:

a) перевести координати точки з простору фото *(I, J)* в простір логічних координат *(XY)* задати координату *Z (Z = Z + dZ)*;
b) виконати перерахунок координат точки з головною СК *(X, Y, Z)* в допоміжну СК *(x, y)*;
c) перевести координати точки з простору логічних координат *(x, y)* в простору фото *(I1, J1)*;
d) перевірити на відповідність за кольором точки головного зображення *(I, J)* і допоміжного *(I1, J1)*.

Цифрове зображення зберігається у вигляді прямокутної матриці, елементи якої несуть інформацію про колір елементарних ділянок зображення (пікселів), а номери рядка I і стовпці J елемента визначають його положення в матриці.

Розрізняють дві прямокутних системи координат цифрового зображення (рис.3.4): ліву, початком якої є лівий верхній піксель; праву - початком якої є лівий нижній піксель.

Ліва СК прийнята під час запису зображень в файл у всіх форматах і використовується в більшості програм по обробці зображень. В фотограмметрії традиційно застосовується права СК для аналізу знімка, і багато сучасні цифрові фотограмметричні системи використовують саме цю систему координат. Для переходу з однієї СК в іншу досить виконати перетворення *J = H - J*.

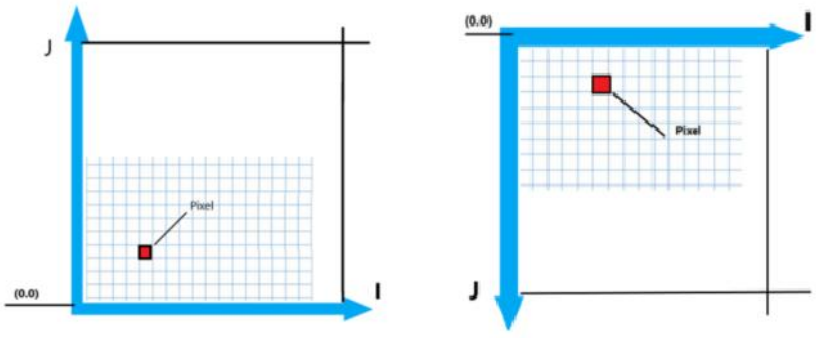


Рисунок 3.4 - Два різновиди прямокутних системи кординат цифрового зображення

У логічній системі координат (рис.3.6) вершини нормалізуються, тобто задовольняють таким умовам:

a) центр координат перенесений в центр екрану;
b) напрямок осі *Y* вгору;
c) координати (від -1 до +1) співвіднесені з шириною (*Width*) і висотою вікна (Height).

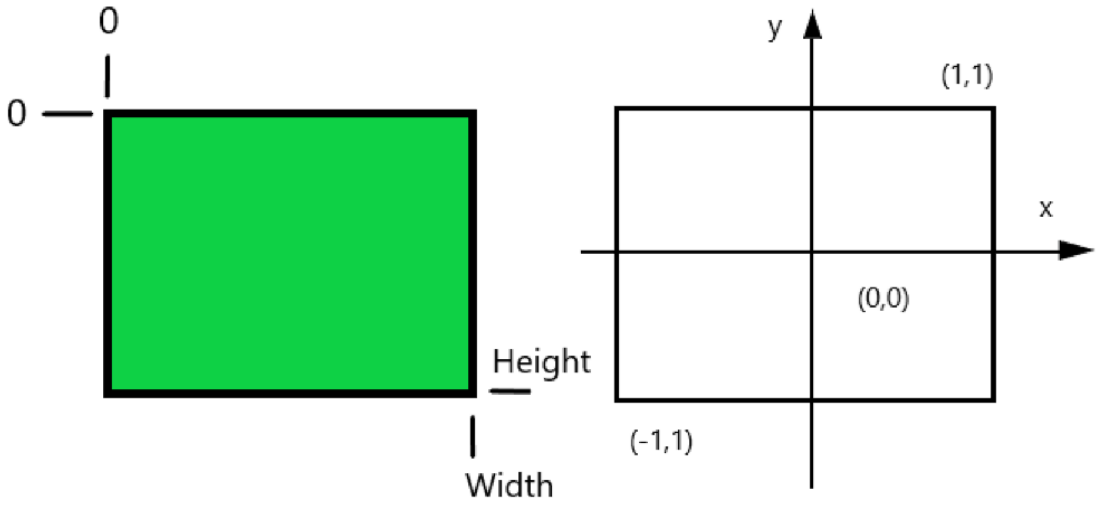


Рисунок 3.6 - Перехід від цифрової до логічної системи координат

Співвідношення для переходу з СК цифрового зображення в нормалізовану СК і назад:

$$X = 2 * \frac{I}{W-1}$$

$$Y = 1 - 2 * \frac{J}{H};\ Y = 1 - 2 * \frac{J}{H-1}$$

$$Z < 0 < 1 \qquad (3.10)$$

$$I1 = W * \frac{(x+1)}{2}$$

$$J1 = -H * \frac{(y-1)}{2};\ J1 = H * \frac{(y-1)}{2}$$

Визначаються різниці сигналів між передбачуваними відповідними точками:

$$dR = abs\,(rgb1\,[I1]\,[J1]\,.rgbRed - rgb\,[I]\,[J]\,.rgbRed)$$
$$dG = abs\,(rgb1\,[I1]\,[J1]\,.rgbGreen - rgb\,[I]\,[J]\,.rgbGreen) \quad (3.11)$$
$$dB = abs\,(rgb1\,[I1]\,[J1]\,.rgbBlue - rgb\,[I]\,[J]\,.rgbBlue)$$

Відповідність між точками встановлено, якщо виконується комбіноване умова:

$$(dR < dmin \text{ і } dG < dmin \text{ і } dB < dmin) \quad (3.12)$$

При виконанні цієї умови одночасно визначається і значення координати Z шуканої точки.

3.4 Тестування алгоритму

Створено проект WinAPI додаток з підключенням бібліотеки glut32 (додаток А). Тестові зображення розміщуються в папці (Data) проекту програми - де і вихідні файли проекту. Вони завантажуються з функції WndProc в файлі main.cpp. Початкове положення ДСК xyz щодо ГСК XYZ визначається через матриці повороту щодо осей ГСК - *gamma* (*Y*), *beta* (*X*) і *alpha* (*Z*) встановлюються у функціях класу Matrix:

SetTranslationMatrix_4 (); // gamma (Y)

SetTranslationMatrix_5 (); // beta (X)

SetTranslationMatrix_6 (); // alpha (Z)

У функції Engine::GetZ (додаток А) реалізується алгоритм відновлення координати *Z*. Тут задаються кути *alpha* (*Z*), *beta* (*X*) і *gamma* (*Y*), які задіяні в системі рівнянь (3.9). На точність тривимірної реконструкцій впливають 2 параметра: dz = 0.01; - крок ітерації по координаті Z; dmin - точність збігу по колірних параметрами. Для тесту с використанням зображень отримані з допомогу камери або захоплення зображення екрану, значення параметра "точність відповідності по кольоровому параметру" дорівнює (dmin = 15 – 80).

Набір вхідних зображень отриманий з бібліотеки тривимірних об'єктів Paint 3D. У редакторі тривимірної моделі встановлювалися параметри об'єкта щодо осі координат (X, Y, Z), шляхом зміни положення об'єкта в тривимірному просторі. Отримане положення об'єкта конвертується в двомірне зображення.

Таблиця 3.1 Параметри вхідних зображень

Номер рисунку	Alpha	Beta	Gamma	Інтераціія	Колірної параметр
Тест 1	00.00	30.00	30.00	0.01	80.00
Тест 2	00.00	32.00	20	0.01	70.00
Тест 3	00.00	70	50	0.01	30.00
Тест 4	10.00	30.00	10.00	0.01	60

Тестове зображення номер один, створюються під заданими кутами alpha, beta і gamma допоміжної СК щодо головної СК. Для першого зображення кути alpha = 0.0, beta = 30.00 і gamma = 30.00, dz = 0.01; - крок ітерації по координаті Z. dmin – 80; точність відповідності по кольоровому параметру. Тестування алгоритму проводиться в 4 етапи:

1) Завантаження головного тестового зображення номер 1;

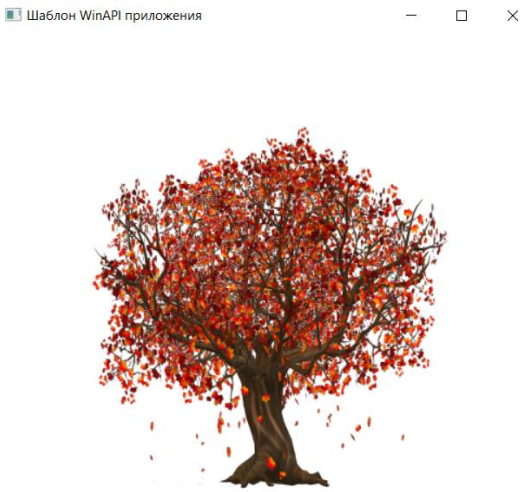


Рисунок 3.7 - Головне тестове зображення номер один

2) Завантаження допоміжного тестового зображення номер 1;

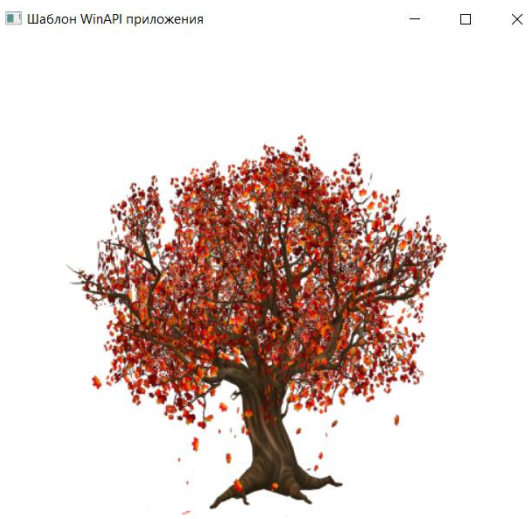


Рисунок 3.8 - Допоміжне зображення номер один

3) Перевірка на відповідність за кольором точки головного тестового зображення номер один і допоміжного тестового зображення номер один.

Рисунок 3.9 - Перевірка на відповідність за кольором

4) Визначення ітерації по координаті Z.

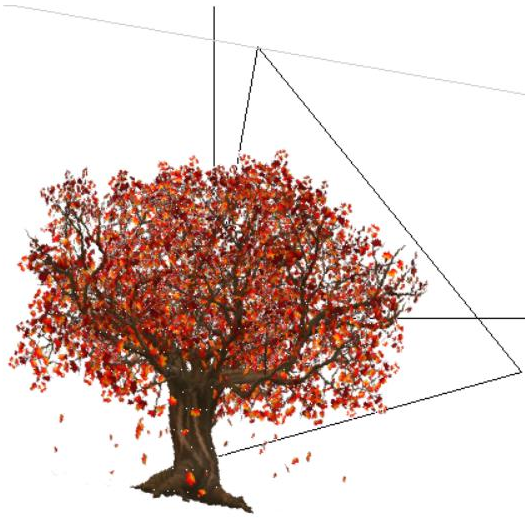


Рисунок 3.10 - Обробка зображення при Z = 0

5) Побудова вершин точок по знайденим відповідностям. Проектування хмари точок в тривимірному просторі.

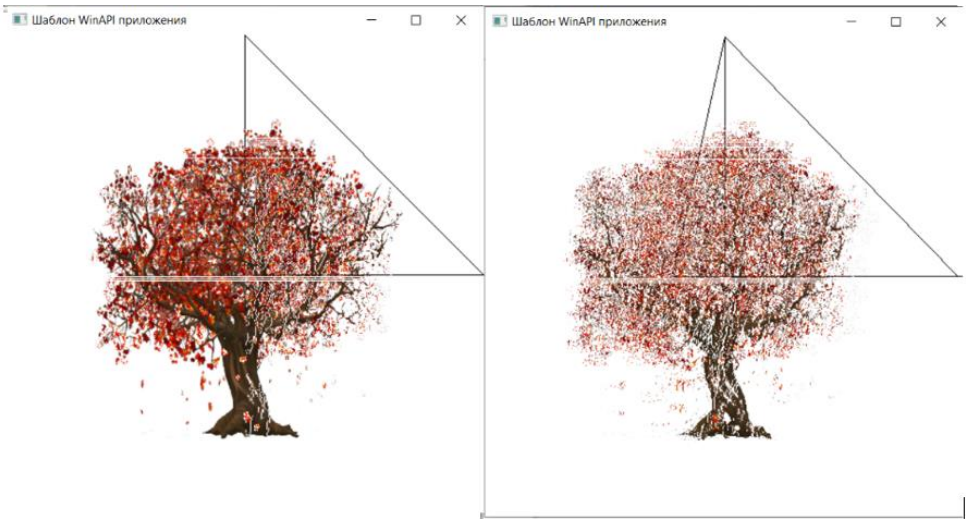


Рисунок 3.10 - Результат тривимірної реконструкцій із хмари точок

Переміщення об'єкта в тривимірному просторі відбувається по переміщень камери щодо осі реконструйованого тривимірного об'єкту.

3.5 Аналіз запропонованого алгоритму

Для аналізу запропонованого алгоритму потрібно визначити наступне: обчислювальну складність алгоритму; точність реконструюється моделі; швидкість центрального процесора при виконанні перевірки на відповідність за кольором точок; швидкість центрального процесора при виконанні визначення ітерації по координаті Z; швидкість центрального процесора при виконанні проектування хмари точок в тривимірному просторі.

Обчислювальна складність алгоритму є O(WHN), де W - ширина зображення, H - висота, а N - кількість відповідностей кольорових точок на головному і допоміжному зображенні.

Для знаходжень точності реконструюються об'єкта потрібно провести попіксельне порівняння отриманого об'єкта з вхідним зображенням.

Знаходження швидкості виконання перерахованих вище дій центральним процесором в мікросекундах, виконується за допомогою вбудованої бібліотеки "time", "ctime" з використанням функції clock(). Для тестування використовувався комп'ютер з процесором Intel® Core™ i7-8550U CPU @ 1.80GHz 1.99GHz і графічним адаптером Radeon™ 520 з оперативною пам'яттю 8 Гб.

Таблиця 3.2 Аналіз роботи алгоритм

Номер тесту	Обчислювальна складність	Точність	Час роботи в мікросекундах (CPU)			
			Відповідність за кольором	Ітерація	Проектування хмари точок	Загальна сума часу
Тест 1	O(WHN)	92.73%	0.234	0.283	0.632	1.15
Тест 2	O(WHN)	91.26%	0.241	0.27	0.65	1.161
Тест 3	O(WHN)	89.53%	0.273	0.29	0.67	1.233
Тест 4	O(WHN)	86.08%	0.28	0.312	0.693	1.29

Виходячи з аналізу проведених тестів в середньому точність побудованої тривимірної моделі близько 90%. В середньому швидкість виконання алгоритму становить 1 секунда.

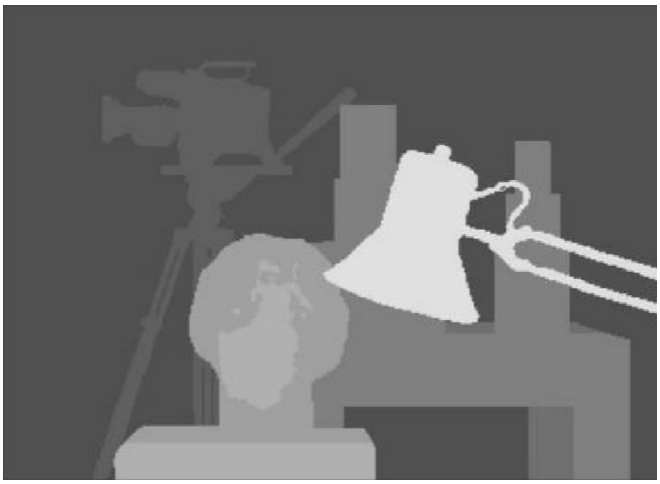
3.6 Порівняльний аналіз алгоритмів

Для проведення порівняльного аналізу алгоритму були обрані існуючі метод і алгоритм. Першим є метод локального пошуку, оскільки складність методу відповідає запропонованому алгоритму, а так само даний метод є комбінованим з методом WTA [3]. Результат роботи методу на рисунку 3.11. Наступний алгоритм "Евристичний алгоритм сегментації хмари точок" являє собою відновлення поверхні з хмари точок тривимірного простору [4]. Результат роботи алгоритму на рисунку 3.12.

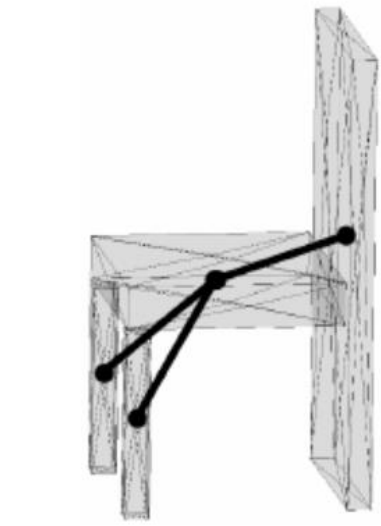
Таблиця 3.2 Порівняльний аналіз алгоритмів і методів

Назва методу	Обчислювальна складність	Точність
Запропонований алгоритм	O(WHN)	90%
Локальний метод пошуку.	O(WHD)	73%
Евристичний алгоритм сегментації хмари точок	$O(n * logn * \frac{2\pi}{k} + \frac{n}{\varepsilon^3})$	90%

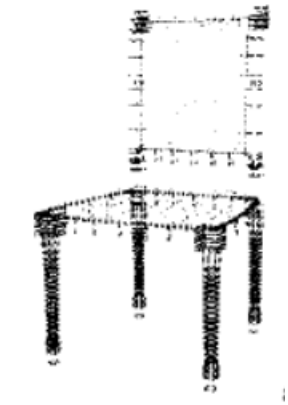
3.11 - Початкове зображення для методу локального пошуку [3]



3.12 - Результат роботи методу локального пошуку [3]



3.13 - Початкове зображення для евристичного алгоритму сегментації хмар точок [4]



3.14 - Результат роботи евристичного алгоритму сегментації хмар точок [4]

Виходячи з порівняльного аналізу можна зробити висновок що метод локального пошуку має невисоку обчислювальну складність, але показує в 74% відсотка точність побудови моделі. Евристичний алгоритм сегментації хмари точок показує хорошу точність відновлення моделі але має високу обчислювальну складність, оскільки алгоритм зіставляє вершину графа з кожним отриманим сегментом хмари точок.

3.7 Висновок до третього розділу

У роботі описаний алгоритм тривимірних реконструкції по набору зображень і формує на їх основі хмара точок об'єкта. Алгоритм програмно реалізований на мові C ++ під операційну систему Windows 10. Алгоритм побудований на основі рішення однієї з безлічі завдань афінних перетворень в однорідних координатах. Алгоритм описаний тільки для найпростішого випадку ортогонального проектування, коли взаємне положення камер визначається тільки 3-ма кутовими параметрами. Так алгоритм може бути узагальнено для випадку, коли взаємне положення камер визначається великою кількістю параметрів, включаючи апарат центрального проектування та враховуючи не тільки обертання, але і переміщення пристрою в просторі.

Для тестування алгоритму було підготовлено ряд тестових зображень з різними вихідними параметрами. Алгоритм має обчислювальну складність O(WHN), в середньому точність побудованої моделі дорівнює 90 відсотків, а швидкість роботи в середньому дорівнює 1 секунда. Порівняльний аналіз показав що запропонований алгоритм не поступається існуючим методам і алгоритмам тривимірної реконструкції. Для того, щоб поліпшити якість тривимірної реконструкції необхідно вирішити ряд завдань попередньої обробки зображень, відділення фону, врахувати втрати якості, пов'язані з стисненням зображень. Але є можливість втрати точності збігу по кольоровому параметру якщо кути повороту камери були задані не правильно. Алгоритм в подальшому може бути модифікований методом тріангуляції. Де тріангуляція це розбиття геометричного об'єкта на трикутники, які будуються по ближнім точкам.

3.8 Перелік джерел посилань до розділу 3

1) Кути Ейлера. Посилання: https://ru.wikipedia.org/wiki/Углы_Эйлера.

2) Матриці базових перетворень 3D простору. Посилання: https://api-2d3d-cad.com/g_transform/

3) А.В. Аргутін. Аналіз швидкодії і обчислювальної складності алгоритмів тривимірної реконструкції. Посилання: https://cyberleninka.ru/article/n/analiz-bystrodeystviya-i-vychislitelnoy-slozhnosti-algoritmov-3d-rekonstruktsii-s-tochki-zreniya-ih-primenimosti-na-protsessorah-s

4) Гасилов А.В. Фролов О.І. Евристичний алгоритм сегментації хмари точок. Посилання: https://cyberleninka.ru/article/n/evristicheskiy-algoritm-segmentatsii-oblaka-tochek/viewer

РОЗДІЛ 4

ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В даному розділі проведено аналіз потенційних небезпечних причин. Розглянуті заходи, які дозволяють забезпечити гігієну праці та виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики. Так як завданням магістерської дипломної роботи є розробка алгоритм відновлення тривимірної графічної моделі на основі двовимірних зображень, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп'ютера та фото обладнання.

4.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Основними характеристиками персонального комп'ютера є наступні:

1) робоча напруга U=+220В +-5%;
2) робочий струм I=2А;
3) споживана потужність P=350 Вт.

Роботу користувача розробленої підсистеми слід віднести до категорії Іа (легкі фізичні роботи) відповідно до даної категорії відносяться всі види діяльності, які виконуються сидячи й не вимагають фізичного напруження.

При експлуатації даного програмного продукту відповідно існують наступні небезпечні й шкідливі виробничі фактори:

1) фізичні:
   a) підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини;
   b) підвищена або знижена вологість повітря;
   c) підвищена або знижена рухливість повітря;
   d) підвищений рівень статичної електрики;
   e) підвищена напруженість електричного й магнітного полів;
   f) відсутність або нестача природного світла;
   g) знижена освітленість робочої зони;

h) підвищений рівень шуму на робочому місці;

i) підвищений рівень електромагнітного випромінювання;

j) знижена контрастність.

2) психофізіологічні:

a) фізичні перевантаження: статичні та динамічні;

b) нервово-психічні перевантаження: розумове перенапруження, монотонність праці, перенапруження аналізаторів та емоційні перевантаження.

4.2 Електробезпека

Основним небезпечним фактором при роботі з персональним комп'ютером є небезпека ураження людини електричним струмом, яка посилюється тим, що органи чуття людини не можуть на відстані виявити наявність електричної напруги на устаткуванні.

Проходячи через тіло людини, електричний струм справляє на нього складну дію, що є сукупністю термічної (нагрів тканин і біологічних середовищ), електролітичної (розкладання крові і плазми) і біологічної (подразнення і збудження нервових волокон та інших органів тканин організму) дій.

Тяжкість ураження людини електричним струмом залежить від цілого ряду чинників:

1) значення сили струму;
2) електричного опору тіла людини і тривалості протікання через нього струму;
3) роду і частоти струму;
4) індивідуальних властивостей людини і навколишнього середовища.

Відповідно до, приміщення для персонального комп'ютера відноситься до приміщень без підвищеної небезпеки, тобто до приміщень, у яких відсутні умови, що створюють підвищену або особливу небезпеку. Небезпека ураження електричним струмом існує всюди, де використовуються електроустановки, тому приміщення без підвищеної небезпеки не можна назвати безпечними.

Електробезпека забезпечується:

1) відповідною конструкцією електроустановок;
2) застосуванням технічних способів і засобів захисту;
3) організаційними і технічними заходами.

Конструкція електроустановок відповідає умовам їхньої експлуатації і забезпечує захист персоналу від дотику до струмоведучих частин.

Основними технічними способами й засобами захисту від ураження електричним струмом, які використовуються окремо або в поєднанні один з одним, є:

1) захисне заземлення;
2) занулення;
3) вирівнювання потенціалів;
4) мала напруга;
5) електричне розділення мереж;
6) захисне відключення;
7) ізоляція струмоведучих частин;
8) компенсація струмів замикання на землю;
9) захисні пристрої;
10) попереджувальна сигналізація, блокування, знаки безпеки;
11) ізолюючі захисні і запобіжні пристосування.

Основними технічними способами і засобами захисту від ураження електричним струмом, що передбачаються в даному дипломному проекті, є:

1) захисне заземлення;
2) занулення;
3) захисне відключення;
4) ізоляція струмоведучих частин.

Занулення в комплексі із захисним відключенням зменшує напругу дотику і обмежує час, в перебігу якого людина, торкнувшись до корпусу, може потрапити під дію напруги.

4.3 Гігієнічні вимоги до параметрів виробничого середовища

4.4 Мікроклімат

Трудова діяльність людини завжди протікає в певних метеорологічних умовах, які визначаються поєднанням температури повітря, швидкості його руху і відносної вологості, тиском і тепловим випромінюванням від нагрітих поверхонь. Оскільки експлуатація проектованого програмного засобу відбувається в приміщенні, то ці показники в сукупності (за

винятком тиску) називаються мікрокліматом виробничого приміщення. В даний час основним нормативним документом нормалізації мікроклімату є ДСН 3.3.6.042-99 [1].

Тяжкість праці характеризує сукупну дію всіх елементів, що складають умови праці, на працездатність людини, його здоров'я, життєдіяльність і відновлення робочої сили. У такому представлені поняття тяжкості праці однаково застосовне як до розумової, так і до фізичної праці. Відповідно ДСН 3.3.6.042-99 [1] тяжкість роботи персоналу, відноситься до легкої категорії 1б (роботи, що виконуються сидячи, не вимагаючи систематичного фізичного напруження і перенесення важкостей) ДСН 3.3.6.042-99 [1]. Загальні санітарно-гігієнічні вимоги до повітря робочої зони. Оптимальні норми мікроклімату в робочій зоні, забезпечувані для робіт легкої категорії 1б приведені в табл. 4.3.1 [1].

У приміщенні, де знаходяться персональний комп'ютер, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти) і установки кондиціонера Брівень якого відповідає К-2000. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП (30 кубічних метрів на годину на одного працюючого).

Таблиця 4.3.1 - Оптимальні норми мікроклімату

Період Року	Температура, °С	Відносна вологість, %	Швидкість руху повітря, м/с, не більш
Холодний і перехідної	21 – 23	60 - 40	0,1
Теплий	22 – 24	60 - 40	0,2

Для захисту від електромагнітного випромінювання передбачаються наступні заходи:

1) застосування нових плазмових моніторів;

2) віддалення робочого місця не менше, ніж на 0,4 - 0,5 м, оскільки напруженість електричного поля зменшується при віддаленні від джерела поля;

3) встановлення раціональних режимів роботи персоналу (обмеження часу перебування);

4) раціональне розміщення в робочому приміщенні устаткування, що випромінює електромагнітну енергію.

Оскільки рівень шуму не перевищує гранично допустимих величин, які встановлені санітарними нормами, заходи для зниження шуму не проводяться.

Для зниження стомлюваності обслуговуючого персоналу в приміщеннях, де розташовані обчислювальні засоби, передбачається використовувати спокійні колірні поєднання і покриття, що не дають відблисків.

4.3.2 Освітлення

Світло є природною умовою існування людини. Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Хороше освітлення діє тонізуюче, створює гарний настрій, покращує протікання основних процесів вищої нервової діяльності.

Збільшення освітленості сприяє поліпшенню працездатності навіть в тих випадках, коли процес праці практично не залежить від зорового сприйняття. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, виникає потенційна небезпека помилкових дій і нещасних випадків.

У проекті, що розробляється, передбачається використовувати суміщене освітлення. У світлий час доби використовуватиметься природне освітлення приміщення через віконні отвори, в решту часу використовуватиметься штучне освітлення. Штучне освітлення створюється газорозрядними лампами.

Штучне освітлення [7] в робочому приміщенні передбачається здійснювати з використанням люмінесцентних джерел світла в світильниках загального освітлення, оскільки люмінесцентні лампи мають високу потужність (80 Вт), тривалий термін служби (до 10000 годин), спектральний складом випромінюваного світла, близький до сонячного. При експлуатації персонального комп'ютера виконується зорова робота IV в розряду точності (середня точність). При цьому нормована освітленість на робочому місці (Ен) рівна 200 лк. Джерелом природного освітлення є сонячне світло. У приміщенні, де розташовані персональний комп'ютер передбачається природне бічне освітлення, рівень якого відповідає ДБН В.2.5-28:2018 [7]. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення. Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати

приміщення, довжина якого складає 8 м, ширина 5 м, світильниками ЛПО2П, оснащеними лампами типу ЛБ (дві по 80 Вт) з світловим потоком 5400 лм кожна.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (4.1):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M} \tag{4.1}$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300лк;

S– освітлювана площа, м2; S = 24 м2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575;

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.1), отримуємо:

$$n = \frac{300 * 24 * 1.1 * 1.5}{5400 * 0.575 * 2} = 1.91$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

4.4 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Застосовують різні електричні захисні засоби від ураження струмом:

1) ізолюючі - ізолюють людину від струмоведучих або заземлених частин, а так-же від землі;

2) основні - володіють ізоляцією, здатної довго витримувати робочу напругу електроустановки і тому ними дозволяється стосуватися струмоведучих частин, що знаходяться під напругою;

3) запобіжні - володіють ізоляцією нездатною витримати робоча напруга електроустановки, і тому вони не можуть самостійно захищати людину від ураження струмом цим напругою. Їх значення - посилити захисні дії основних і ізолюючих засобів, разом з якими вони повинні застосовуватися, причому при використання основних захисних засобів достатньо застосування одного заходи захисного засобу [18,19].

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Відповідно до класифікації приміщень за ступенем небезпеки ураження електричним струмом [3], приміщення в якому проводяться всі роботи належить до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 36 В, 220 В, і 360 В. Опір контуру заземлення повинен мати не більше 4 Ом.

Розрахунок проводять за допомогою методу коефіцієнта використання (екранування) електродів. Коефіцієнт використання групового заземлення η - це ставлення чинної провідності цього заземлення до найбільш можливої його провідності при нескінченно великих відстаней між його електродами. Коефіцієнт використання вертикальних заземлювачів $\eta_в$ у залежності від розміщення заземлювачів і їх кількості знаходиться в межах 0,4 ... 0,99. Взаємну екрануючого дії горизонтального заземлювача (сполучної смуги) враховують за допомогою коефіцієнта використання горизонтального заземлювача $\eta_с$.

Послідовність розрахунку:

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{\text{шт.з.}} = \frac{R_{\text{д}} \cdot R_{\text{пр.з.}}}{R_{\text{пр.з.}} - R_{\text{д}}} \qquad (4.2)$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.}=R_д$.

Підставивши числові значення в формулу (4.2), отримуємо:

$$R_{\text{шт.з.}} = \frac{4 \cdot 40}{40-4} \approx 4\text{Ом}$$

1) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом • м. Приблизне значення питомої опору глини приймаємо ρ=40 Ом•м (табличне значення).

2) Розрахункова питомий опір ґрунту, $\rho_{розр}$, Ом•м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом•м по формулі:

$$\rho_{\text{розр..}} = \psi \cdot \rho, \tag{4.3}$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів і кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в}$=1,7 і горизонтальних $\rho_{розр.г}$=5,5 Ом•м.

$$\rho_{\text{розр.в}} = 1{,}7 \cdot 40 = 68 \text{ Ом•м}$$

$$\rho_{\text{розр.г}} = 5{,}5 \cdot 40 = 220 \text{ Ом•м}$$

1) Розраховується опір розтікання струму вертикального заземлення $R_в$, Ом, по (4.4).

$$R_{\text{в}} = \frac{\rho_{\text{розр.в}}}{2 \cdot \pi \cdot l_{\text{в}}} \cdot \left(ln \frac{2 \cdot l_{\text{в}}}{d_{\text{ст}}} + \frac{1}{2} \cdot ln \frac{4 \cdot t + l_{\text{в}}}{4 \cdot t - l_{\text{в}}} \right), \tag{4.4}$$

де lв – довжина вертикального заземлювача (для труб - 2−3 м; lв=3 м); dст – діаметр стрижня (для труб - 0,03−0,05 м; dст=0,05 м); t – відстань від поверхні землі до середини заземлювача, яка визначається за формулою (4.5):

$$t = h_{\text{в}} + \frac{l_{\text{в}}}{2}, \tag{4.5}$$

де $h_{\text{в}}$ – глибина закладення вертикальних заземлювачів (0,8 м); тоді

$$t = 0{,}8 + \frac{3}{2} = 2{,}3 \text{ м}$$

$$R_{\text{в}} = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(ln \frac{2 \cdot 3}{0{,}05} + \frac{1}{2} \cdot ln \frac{4 \cdot 2{,}3 + 3}{4 \cdot 2{,}3 - 3} \right) = 18{,}5 \text{Ом}$$

4) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_{\text{в}}$:

$$n = \frac{2 \cdot R_{в}}{R_{д}} = \frac{2 \cdot 18{,}5}{4} = 9{,}25$$

І визначається коефіцієнт використання вертикальних електродів групового заземлення без урахування впливу сполучної стрічки $\eta_{\text{в}}$ =0,57 (табличне значення).

5) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_{\text{в}}$, шт:

$$n_{\text{в}} = \frac{2 \cdot R_{\text{в}}}{R_{\text{д}} \cdot \eta_{\text{в}}} = \frac{2 \cdot 18{,}5}{4 \cdot 0{,}57} = 16{,}2 \approx 16$$

6) Визначається довжина сполучної стрічки горизонтального заземлювача $l_{\text{с}}$, м:

$$l_{\text{с}} = 1{,}05 \cdot L_{\text{в}} \cdot (n_{\text{в}} - 1), \tag{4.6}$$

де $L_{\text{в}}$ – відстань між вертикальними заземлювачами, (прийняти $L_{\text{в}}$ = 3м);

$n_в$ – необхідну кількість вертикальних заземлювачів.

$$l_с = 1{,}05 \cdot 3 \cdot (16 - 1) \approx 48м$$

8) Визначається опір розтіканню струму горизонтального заземлювача (сполучної стрічки) $R_г$, Ом:

$$R_г = \frac{\rho_{розр.г}}{2 \cdot \pi \cdot l_с} \cdot ln \frac{2 \cdot {l_с}^2}{d_{см} \cdot h_г}, \quad (4.7)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ =0,95b, b = 0,15 м; $h_г$ – глибина закладення горизонтальних заземлювачів (0,5 м); $l_с$ - довжина сполучної стрічки горизонтального заземлювача $l_с$, м.

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot ln \frac{2 \cdot 48^2}{0{,}95 \cdot 0{,}15 \cdot 0{,}5} = 8{,}1$$

9) Визначається коефіцієнт використання горизонтального заземлювача ηс відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання сполучної смуги $\eta_с$=0,3 (табличне значення).

10) Розраховується результуючий опір заземлюючого електрода з урахуванням сполучної смуги:

$$R_{заг} = \frac{R_в \cdot R_г}{R_в \cdot \eta_с + R_г \cdot n_в \cdot \eta_в} \leq R_д. \quad (4.8)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпека будівлі [10], так як виконується умова: $R_{заг}$ <4 Ом, а саме:

$$R_{\text{заг}} = \frac{18{,}5 \cdot 8{,}1}{18{,}5 \cdot 0{,}3 + 8{,}1 \cdot 16 \cdot 0{,}57} = 1{,}9 \leq R_{\text{д}}$$

При виникненні пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як: іскри і дуги коротких замикань; перегрів провідників, резисторів і інших радіодеталей ПЕОМ, від тривалого перевантаження і наявність перехідного опору; іскри при розмиканні і розмиканні ланцюгів; розряди статичної електрики; необережне поводження з вогнем, а також вибухи газоповітряних і пароповітряних сумішей.

Важливу увагу слід звернути на пожежну безпеку підприємства в цілому і окремих його приміщень. У приміщеннях не повинно накопичуватися сміття, непотрібну папір, мотлох та ін. Речі, які не використовуються у виробничому процесі. Наявний вільний аварійний вихід за межі приміщення в разі пожежі, бути передбачені вогнегасники. Вони повинні бути в робочому стані і перевірятися відповідно до норм. У приміщеннях повинна бути пожежна сигналізація, вогнегасник. У разі виникнення пожежі необхідно повідомити в найближчу пожежну частину, убезпечити інших працівників і по можливості прийняти кроки щодо запобігання можливих наслідків та усунення пожежі [3].

4.5 Охорона навколишнього природного середовища

Діяльність за темою магістерської роботи в процесі її виконання впливає на навколишнє природне середовище і регламентується нормами діючого законодавства [25-29].

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на видалення (знешкодження), утилізацію, тощо в ІТ галузі.

В процесі діяльності користувача виникають процеси поводження з відходами ІТ галузі. Види відходів, утворення, яких можливо:

- відпрацьовані люмінесцентні лампи - I клас небезпеки;
- батарейки та акумулятори (малі) -III клас небезпеки;
- змінні носії інформації - IV клас небезпеки;
- макулатура - IV клас небезпеки;
- побутові відходи - IV клас небезпеки.

4.6 Висновок до четвертого розділу

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.

Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Було наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

У двадцять першому столітті актуальною проблемою є забруднення навколишнього середовища. Проблеми з екологією зустрічаються в повсякденному житті і в будь-який діяльності людини, і не є винятком сфери пов'язаний інформаційними технологіями. Важливо розуміти що використання несправного обладнання або неправильної експлуатації, впливає не тільки на здоров'я людини, а також на навколишнє середовище. Так само потрібно сортувати і утилізувати відходи в процесі роботи. Виходячи з вищесказаного можна зробити висновок що дотримуватися нормативних документів охорони праці є обов'язковим.

4.7 Перелік джерел посилань до розділу 4

1. ДСН 3.3.6.042-99 Санітарні норми виробничих приміщень - https://zakon.rada.gov.ua/rada/show/va042282-99 - 01.12.1999р.

2. НПАОП 40.1-1.01-97 Правила безопасной эксплуатации электроустановок - https://zakon.rada.gov.ua/laws/show/z0011-98 - 13.01.1998р.

3. НПАОП 40.1-1.21-98 Правила безпечної експлуатації електроустановок споживачів - https://zakon.rada.gov.ua/laws/show/z0093-98 - 10.02.1998

4. Міністерство енергетики та вугільної промисловості України Наказ від 13.02.2012 № 91 Про внесення змін та
доповнень до Правил технічної експлуатації електроустановок споживачів - https://zakon.rada.gov.ua/laws/show/z0350-12.

5. НАПБ А.01.001-2014. Правила пожежної безпеки в Україні. - https://zakon.rada.gov.ua/laws/show/z0252-15 - 05.03.2015р.

6. ДСанПіН 3.3.2.007-98 «Правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» - https://zakon.rada.gov.ua/rada/show/v0007282-98 - 10.12.1998р.

7. ДБН В.2.5-28:2018 «Природне і штучне освітлення» - https://dbn.co.ua/load/normativy/dbn/dbn_v_2_5_28/1-1-0-1188 - 28.02.2019р.

8. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку - https://zakon.rada.gov.ua/go/va037282-99 - 01.12.1999р.

9. Закон України «Про охорону навколишнього природного середовища» - https://zakon.rada.gov.ua/laws/show/1264-12 - 12.10.2018р.

10. Закон України «Про забезпечення санітарного та епідемічного благополуччя населення» - https://zakon.rada.gov.ua/laws/show/4004-12 - 04.10.2018р.

11. Закон України «Про відходи» - https://zakon.rada.gov.ua/laws/show/187/98-%D0%B2%D1%80 - 01.05.2019р.

12. Закон України «Про охорону атмосферного повітря» - https://zakon.rada.gov.ua/laws/show/2707-12 - 18.12.2017р.

13. Закон України Закон України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру» - https://zakon.rada.gov.ua/laws/show/1809-14 -01.07.2013

ВИСНОВКИ

Метою магістерської атестаційної роботи було аналіз технології відновлення тривмірної графічної інформації на основі двовимірних зображень. Аналіз систем і компонентів обробки зображення, задач відновлення даних в третьому вимірі з двовимірних даних.

В ході магістерської роботи були досягнуті такі результати як:

1) Проведена класифікація методів тривимірної реконструкції.

2) Аналіз останніх публікацій тривимірної реконструкції та аналіз моделей, що використовуються в задачах тривимірної реконструкції.

3) Встановлено що багато методів і алгоритмів реконструкції мають ряд проблем, а самє точність побудова моделі, розпізнавання об'єкта, алгоритми обробки зображення комп'ютерного зору, які не завжди дозволяють реконструювати об'єкти точно чином.

4) Проведено аналіз сучасних алгоритмів і методів тривимірної реконструкцій. Тривимірна реконструкція має безліч методів і алгоритмів. Кожен алгоритм і метод являє собою набір етапів обробки зображення.

5) Проведена оцінка і порівняння методів. Аналіз показав що високу точність дають методи стерео зіставлення.

6) Проведена оцінка комбінування методів. З оцінки можна зробити висновок що, комбінування дозволяє досягти більш високих результатів.

У ході практичної частини роботи були отримані наступні результати:

1) Розроблено алгоритм тривимірної реконструкції з хмари точок на основі методу стерео зіставлення.

2) На основі запропонованого алгоритму було розроблено програмне забезпечення тривимірної реконструкцій на мові C++.

3) Проведено тестування алгоритму. Виходячи з аналізу проведених тестів в середньому точність побудованої тривимірної моделі близько 90%. В середньому швидкість виконання алгоритму становить 1 секунда.

4) Аналіз запропонованого алгоритму. Де аналіз покозав що обчислювальна складність алгоритму є O(WHN), де W - ширина зображення, H - висота, а N - кількість відповідностей кольорових точок на головному і допоміжному зображенні.

5) Проведено порівняльний аналіз алгоритмів. Де аналіз показав що запропонований алгоритм не поступається існуючим методам і алгоритмам тривимірної реконструкції.

Додаток А.

Результати тривимірної реконструкції тестових зображень

Тестове зображення номер 2

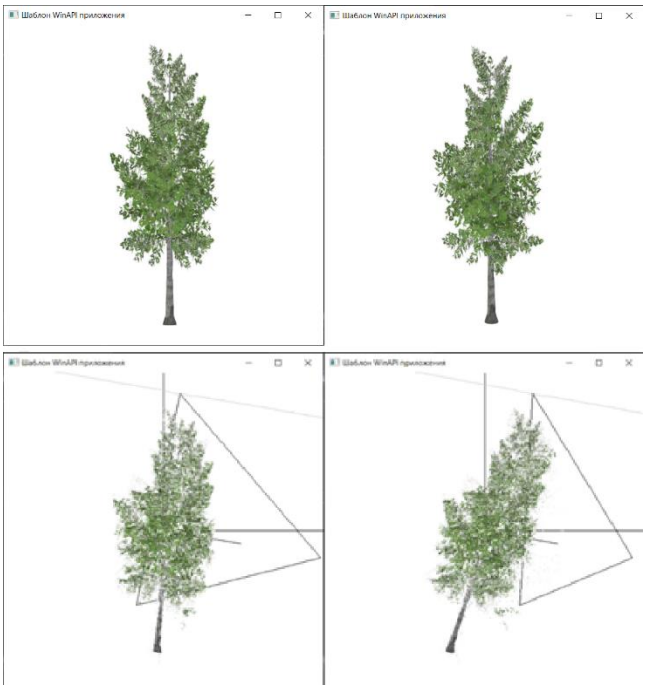


Рисунок 3.13 - Результат тривимірної реконструкцій тествого зображення номер 2

Тестове зображення номер 3

Рисунок 3.14 - Результат тривимірної реконструкцій тествого зображення номер 3

Тестове зображення номер 4

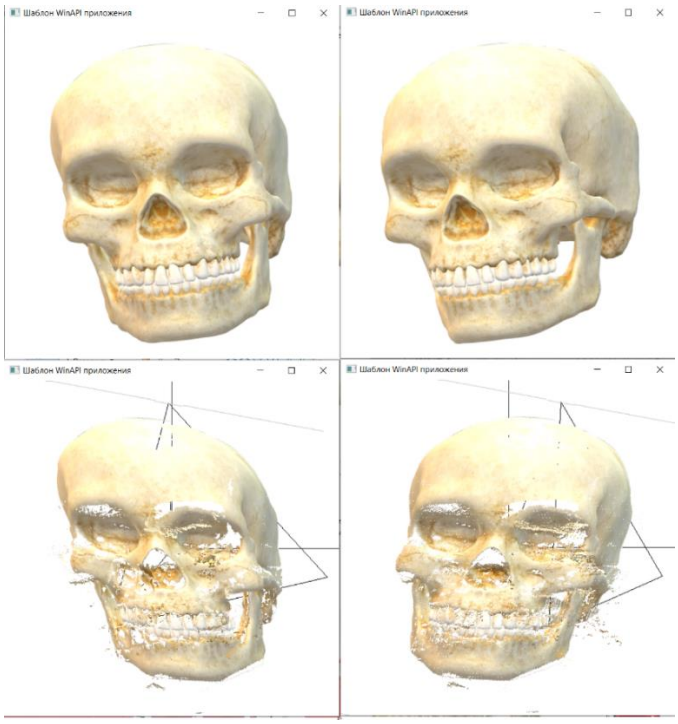


Рисунок 3.15 - Результат тривимірної реконструкцій тествого зображення номер 4

Додаток Б.

Програмна реалізація алгоритму тривимірної реконструкції на мові С ++

Action.cpp

Даний код відповідає за дії переміщення камери по осі тривимірного об'єкту.

```
1. #include "action.h"
2. void Action::InitAction(double x, double y) {
3. old_mouse.set(x, y);}
```

Повертає систему координат, спираючись на поточний і попереднє положення миші

```
4. void Action::Rotate(double x, double y) {
5. vec new_mouse(x, y);
6. vec_float sina, cosa;
7. sina = old_mouse.unit() ^ new_mouse.unit();
8. cosa = old_mouse.unit() * new_mouse.unit();
9. Matrix Rot;
10. Rot.SetRotationMatrixbySinCos(sina, cosa);
11. CurrentMatrix.MultiplyMatrices(Rot);
12. old_mouse = new_mouse;}
```

Перемістити систему координат, спираючись на поточний і попереднє положення миші

```
13. void Action::Translate(double x, double y) {
14. vec new_mouse(x, y);
15. vec delta = new_mouse - old_mouse;
16. Matrix Tr;
17. Tr.SetTranslationMatrix(delta.x, delta.y);
18. CurrentMatrix.MultiplyMatrices(Tr);
19. old_mouse = new_mouse;}
20. void Action::Transform_0() {
21. Matrix Tr;
22. Tr.SetTranslationMatrix_0();
23. CurrentMatrix.MultiplyMatrices(Tr);}
24. void Action::Transform_3() {
25. Matrix Tr;
26. Tr.SetTranslationMatrix_3();
27. CurrentMatrix.MultiplyMatrices(Tr);
28. Tr.SetUnit();}
```

```
29. void Action::Transform_4() {
30. Matrix Tr;
31. Tr.SetTranslationMatrix_4();       //  gamma (Y)
32. CurrentMatrix.MultiplyMatrices(Tr);
33. Tr.SetTranslationMatrix_5();       //  beta (X)
34. CurrentMatrix.MultiplyMatrices(Tr);
35. Tr.SetTranslationMatrix_6();       //  alpha (Z)
36. CurrentMatrix.MultiplyMatrices(Tr);}
37. void Action::Transform_5() {
38. Matrix Tr;
39. Tr.SetTranslationMatrix_7();
40. CurrentMatrix.MultiplyMatrices(Tr);
41. Tr.SetTranslationMatrix_6();
42. CurrentMatrix.MultiplyMatrices(Tr);}
```

Viewport.cpp

Даний код задає параметри вікна програми.

```
1. #include "viewport.h"
```

Здається розмір вікна

```
2. void Viewport::SetWindowSize(int _Width, int _Height) {
3. Width = _Width;
4. Height = _Height;}
```

Виконує перетворення з координат [-1, 1] x [-1, 1] в координати екрана, з урахуванням відступів.

```
5. _Point Viewport::T(_Point point) {
6. _Point TPoint;
7. TPoint.x = Margin + (1.0 / 2) * (point.x + 1) * (Width - 2 * Margin);
8. TPoint.y = Margin + (-1.0 / 2) * (point.y - 1) * (Height - 2 * Margin);
9. TPoint.z = Margin + (1.0 / 2) * (point.z + 1) * (Width - 2 * Margin);
10. return TPoint;}
```

Виконує перетворення з координат [-1, 1] x [-1, 1] в координати екрана, з урахуванням відступів.

```
11. _Point Viewport::T_inv(_Point point) {
12. _Point TPoint;
13. TPoint.x = double(point.x - Margin) / (1.0 / 2) / (Width - 2 * Margin) - 1;
14. TPoint.y = double(point.y - Margin) / (-1.0 / 2) / (Height - 2 * Margin) + 1;
15. TPoint.z = double(point.z - Margin) / (1.0 / 2) / (Width - 2 * Margin) - 1;
16. return TPoint;}
17. Viewport::Viewport() {
18. SetMargin();}
```

Встановлює відступ по краях екрану.

```
19. void Viewport::SetMargin(int _Margin) {
20. Margin = _Margin;}
```

Matrix.cpp

Даний код відповідає за перетворення матриць

```
1. #include <math.h>
2. #include <memory.h>
3. #include "matrix.h"
4. #include "geometry.h"
5. bool w;
6. void SetW(bool _w) { w = _w; }
```

Конструктор, инициализирует матрицю одиничної (матриця тотожного перетворення).

```
7. Matrix::Matrix() {
8. SetUnit();}
```

Поточна матриця стає одиничною.

```
9. void Matrix::SetUnit() {
10. memset(data, sizeof(data), 0);
11. for (int i = 0; i < 4; i++) {
12. for (int j = 0; j < 4; j++) {
13. data[i][j] = 0.0;}}
```

```
14. for (int i = 0; i < 4; i++) {
15. data[i][i] = 1.0;}}
```

Встановлює поточну матрицю в якості матриці обертання на кут alpha, заданий косинусом і синусом

```
16. void Matrix::SetRotationMatrixbySinCos(double sinalpha, double cosalpha) {
17. SetUnit();
18. data[0][2] = sinalpha;
19. data[2][0] = -sinalpha;
20. data[2][2] = cosalpha;
21. data[0][0] = cosalpha;
22. data[0][2] = sinalpha;
23. data[2][0] = -sinalpha;
24. data[2][2] = cosalpha;}
```

Встановлює поточну матрицю в якості матриці обертання на кут alpha.

```
25. void Matrix::SetRotationMatrix(double alpha) {
26. SetRotationMatrixbySinCos(sin(alpha), cos(alpha));}
```

Встановлює поточну матрицю в якості матриці паралельного перенесення на вектор (tx, ty).

```
27. void Matrix::SetTranslationMatrix(double tx, double ty) {
28. SetUnit();
29. double txx = tx * tx;
30. double txy = tx * ty;
31. double tyy = ty * ty;
32. double tsx = sqrt(1 - txx);
33. double tsy = sqrt(1 - tyy);
34. data[0][0] = tsx;
35. data[0][1] = -txy;
36. data[0][2] = tx * tsy;
37. data[1][1] = tsy;
38. data[1][2] = ty;
39. data[2][0] = -tx;
40. data[2][1] = -tsx * ty;
```

```
41. data[2][2] = tsx * tsy; }
42. void Matrix::SetTranslationMatrix_0() {
43. SetUnit();
44. data[2][2] = 0.0; }
45. void Matrix::SetTranslationMatrix_3() {
46. SetUnit();}
47. void Matrix::SetTranslationMatrix_4() { //gm (Y)
48. SetUnit();
49. double gm = 20.0;
50. gm = 3.14*gm / 180;
51. data[0][0] = cos(gm);
52. data[0][2] = sin(gm);
53. data[2][0] = -sin(gm);
54. data[2][2] = cos(gm); }
55. void Matrix::SetTranslationMatrix_5() {
56. SetUnit();
57. double bt = 30.0;
58. bt = 3.14*bt / 180;
59. data[1][1] = cos(bt);
60. data[1][2] = sin(bt);
61. data[2][1] = -sin(bt);
62. data[2][2] = cos(bt); }
63. void Matrix::SetTranslationMatrix_6() { // al (Z)
64. SetUnit();
65. double al = 0.0;
66. al = 3.14*al / 180;
67. data[0][0] = cos(al);
68. data[0][1] = -sin(al);
69. data[1][0] = sin(al);
70. data[1][1] = cos(al);}
71. void Matrix::SetTranslationMatrix_7() {
72. SetUnit();
73. data[1][1] = 0.866;
74. data[1][2] = -0.5;
75. data[2][1] = 0.5;
```

```
76. data[2][2] = 0.866; }
```

Примножує поточну матрицю на матрицю, передану в якості параметра.

```
77. void Matrix::MultiplyMatrices(Matrix &right) {
78. double temp[4][4];
79. double val;
80. memcpy(temp, data, sizeof(data));
81. for (int i = 0; i < 4; i++) {
82. for (int j = 0; j < 4; j++) {
83. val = 0;
84. for (int v = 0; v < 4; v++) {
85. Глобальна система координат.
86. if (w == true) val += temp[i][v] * right.data[v][j];
87. Локальна система координат.
88. else val += right.data[i][v] * temp[v][j];}
89. data[i][j] = val;}}}
```

Перемножує точку, передану в якості параметра на поточну матрицю. При цьому останній рядок матриці не враховується.

```
90. void Matrix::ApplyMatrixtoPoint(_Point &point) {
91. double _x, _y, _z, w;
92. _x = point.x;
93. _y = point.y;
94. _z = point.z;
95. point.x = _x * data[0][0] + _y * data[1][0] + _z * data[2][0] + data[3][0];
96. point.y = _x * data[0][1] + _y * data[1][1] + _z * data[2][1] + data[3][1];
97. point.z = _x * data[0][2] + _y * data[1][2] + _z * data[2][2] + data[3][2];
98. w = data[2][3] * _z + 1;
99. point.x = point.x / w;
100.        point.y = point.y / w;}
```

Main.cpp

Код основного файлу проекту.

```
1. #include "engine.h"
2. const double PI = 3.141592653;
3. LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
4. char szClassName[] = "CG6";
5. char szWindowCaption[] = "CG #6 Mouse Tracking and Rotation";
6. int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow){
7. HWND hWnd;
8. MSG lpMsg;
9. WNDCLASS wc;
```

Заповнюємо структуру класу вікна.

```
10. wc.style = CS_HREDRAW | CS_VREDRAW;
11. wc.lpfnWndProc = WndProc;
12. wc.cbClsExtra = 0;
13. wc.cbWndExtra = 0;
14. wc.hInstance = hInstance;
15. wc.hIcon = NULL;
16. wc.hCursor = LoadCursor(NULL, IDC_ARROW);
17. wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
18. wc.lpszMenuName = NULL;
19. wc.lpszClassName = (LPCWSTR)szClassName;
```

Реєструємо клас вікна.

```
20. if (!RegisterClass(&wc)){
21. MessageBox(NULL, L"Cannot register class", L"Error", MB_OK);
22. return 0;}
```

Створюємо основне вікно програми.

```
23. hWnd = CreateWindow(
24. (LPCWSTR)szClassName, L"Шаблон WinAPI приложения",
25. WS_OVERLAPPEDWINDOW,
26. 100, 100,
```

```
27. 516, 540,
28. (HWND)NULL,
29. (HMENU)NULL,
30. (HINSTANCE)hInstance, NULL);
31. WM_CREATE
32. if (!hWnd){
33. MessageBox(NULL, L"Cannot create main window", L"Error", MB_OK);
34. return 0;}
35. ShowWindow(hWnd, nCmdShow);
36. UpdateWindow(hWnd);
37. while (GetMessage(&lpMsg, NULL, 0, 0)) {
38. TranslateMessage(&lpMsg);
39. DispatchMessage(&lpMsg);}
40. return (lpMsg.wParam);}
41. LRESULT CALLBACK WndProc(HWND hWnd, UINT messg, WPARAM
    wParam, LPARAM lParam){
42. PAINTSTRUCT ps;
43. static RECT Rect;
44. static HDC hdc, hCmpDC;
45. static HBITMAP hBmp;
46. static Action *action;
47. static Engine *engine;
48. static BITMAP bm;
49. static bool r1;
50. static bool r2;
51. static bool r3;
52. static bool r4;
53. static bool w;
54. static int mx;
55. static int my;
56. static int mx1;
57. static int my1;
58. static int im;
59. static RGBQUAD **v1;
60. static RGBQUAD **v2;
```

```
61. switch (messg){
62. case WM_CREATE:
63. engine = new Engine();
```

Виклик конструктора Matrix і ініціалізація матриці (стає одиничної).

```
64. action = new Action();
65. engine->SetAction(action);
66. r1 = true;
67. r2 = true;
68. r3 = true;
69. r4 = true;
70. w = true;
71. SetW(w);
72. im = 0;
73. break;
74. case WM_PAINT:
75. GetClientRect(hWnd, &Rect);
76. hdc = BeginPaint(hWnd, &ps);
77. SetBkColor(hdc, 0xEECCCC);
```

Створення нового контексту для подвійний буферизації.

```
78. hCmpDC = CreateCompatibleDC(hdc);
79. hBmp = CreateCompatibleBitmap(hdc, Rect.right - Rect.left, Rect.bottom - Rect.top);
80. SelectObject(hCmpDC, hBmp);
81. LOGBRUSH br;
82. br.lbStyle = BS_SOLID;
83. br.lbColor = 0xFFFFFF;
84. HBRUSH brush;
85. brush = CreateBrushIndirect(&br);
86. FillRect(hCmpDC, &Rect, brush);
87. DeleteObject(brush);
```

Завантаження зображень.

```
88. if (im == 1) { engine->LoadBMP(hWnd, hBmp, hCmpDC, bm, TEXT("Data/name1.bmp"));}
89. if (im == 2) { engine->LoadBMP(hWnd, hBmp, hCmpDC, bm, TEXT("Data/name2.bmp"));}
```

Отрісовка.

```
90. if (im >= 3){
91. engine->Draw(hCmpDC);
92. engine->ImagePixel(hCmpDC, v1, mx, my);}
```

Проектування на екрані.

```
93. SetStretchBltMode(hdc, COLORONCOLOR);
94. BitBlt(hdc, 0, 0, Rect.right - Rect.left, Rect.bottom - Rect.top, hCmpDC, 0, 0, SRCCOPY);
95. DeleteDC(hCmpDC);
96. DeleteObject(hBmp);
97. hCmpDC = NULL;
98. EndPaint(hWnd, &ps);
99. break;
100.    case WM_SIZE: WM_PAINT WM_SIZE
101.    GetClientRect(hWnd, &Rect);
102.    engine->viewport.SetWindowSize(Rect.right - Rect.left, Rect.bottom - Rect.top);
103.    break;
104.    viewport.Height, viewport.Width;
105.    case WM_ERASEBKGND:
106.    return 1;
107.    break;
108.    case WM_LBUTTONDOWN:
109.    _Point mouse_point;
```

Збережемо координати курсора миші в системі вікна.

```
110.    mouse_point.x = LOWORD(lParam);
111.    mouse_point.y = HIWORD(lParam);
112.    mouse_point = engine->viewport.T_inv(mouse_point);
```

```
113.  action->InitAction(mouse_point.x, mouse_point.y);
114.  break;
115.  case WM_MOUSEMOVE:
116.  if (UINT(wParam) & MK_LBUTTON) {
117.  _Point mouse_point;
118.  mouse_point.x = LOWORD(lParam);
119.  mouse_point.y = HIWORD(lParam);
120.  mouse_point = engine->viewport.T_inv(mouse_point);
121.  if (UINT(wParam) & MK_CONTROL) {
```

Перемістити систему координат, спираючись на поточний і попереднє положення миші.

```
122.  action->Translate(mouse_point.x, mouse_point.y);}else {
123.  action->Rotate(mouse_point.x, mouse_point.y);}
124.  InvalidateRect(hWnd, NULL, FALSE);}
125.  break;
126.  case WM_KEYDOWN:
127.  int KeyPressed;
128.  KeyPressed = int(wParam);
129.  if (KeyPressed == int('W')) { w = (w == true) ? false : true; SetW(w); }
130.  if (KeyPressed == int('0')) { action->Transform_0(); }
131.  if (KeyPressed == int('1')){
```

Зчитування і формування масиву 1-го зображення.

```
132.  if (r1 == true) {
```

Виділяємо пам'ять і читаємо туди файл.

```
133.  v1 = engine->MatrixBMP("Data/name.bmp", mx, my);
134.  r1 = false;}
```

Завантажуємо перше зображення.

```
135.  im = 1;}
136.  if (KeyPressed == int('2')){
```

Зчитування і формування масиву 2-го зображення.

```
137.  if (r2 == true) {
```

Виділяємо пам'ять і читаємо туди файл.

```
138.    v2 = engine->MatrixBMP("Data/triangle1.bmp", mx1, my1);
139.    r2 = false;}
```

Завантажуємо другого зображення.

```
140.    im = 2;}
141.    if (KeyPressed == int('3')){
142.    if (r3 == true) {
```

Фільтрація точок зображень і присвоєння точкам 1-го z = 0.

```
143.    engine->Filter(v1, v2, mx, my);
```

Підготовка відповідної матриці перетворень.

```
144.    action->Transform_4();
145.    r3 = false;}
146.    im = 3;}
147.    if (KeyPressed == int('4')){
```

Визначення відповідних точок 2-х зображень і знаходження для кожної точки 3-й координати

```
148.    if (r4 == true) {
```

Один раз визначити координату Z для точок головного зображення

```
149.    engine->GetZ(v1, v2, mx, my);
```

Масив точок 2-го зображення можна видалити

```
150.    delete v2;
151.    r4 = false;}
152.    im = 4;}
153.    InvalidateRect(hWnd, NULL, FALSE);
154.    break;
155.    case WM_DESTROY:
156.    PostQuitMessage(0);
157.    break;
158.    default:
159.    return (DefWindowProc(hWnd, messg, wParam, lParam));}
```

```
160.    return (0);}
```

Engine.cpp

Даний код реалізує алгоритм тривимірної реконструкцій.

```
1. #include <windows.h>
2. #include "geometry.h"
3. #include "matrix.h"
4. #include "engine.h"
```

Прив'язати об'єкт, який відповідає за дії користувача.

```
5. void Engine::SetAction(Action *_action){
6. action = _action;}
```

Виводить графіку на контекст hdc.

```
7. void Engine::Draw(HDC hdc) {
8. _Point p[12];
9. p[1].x = 1.0;
10. p[1].y = 0.0;
11. p[1].z = 0.0;
12. p[2].x = 0.0;
13. p[2].y = 1.0;
14. p[2].z = 0.0;
15. p[3].x = 0.0;
16. p[3].y = 0.0;
17. p[3].z = 1.0;
```

Локальна система координат.

```
18. p[4].x = 0.0;
19. p[4].y = 0.0;
20. p[4].z = 0.0;
21. p[5].x = 0.5;
22. p[5].y = 0.0;
23. p[5].z = 0.0;
24. p[6].x = 0.0;
```

```
25. p[6].y = 0.5;
26. p[6].z = 0.0;
27. p[7].x = 0.0;
28. p[7].y = 0.0;
29. p[7].z = 0.5;
30. for (int i = 1; i < 8; i++) {
```

Обертання і переміщення здійснюється в логічних координатах.

```
31. action->CurrentMatrix.ApplyMatrixtoPoint(p[i]);
```

Перехід з логічних в віконні координати.

```
32. p[i] = viewport.T(p[i]);}
```

Побудова трикутника.

```
33. MoveToEx(hdc, p[1].x, p[1].y, NULL);
34. LineTo(hdc, p[2].x, p[2].y);
35. LineTo(hdc, p[3].x, p[3].y);
36. LineTo(hdc, p[1].x, p[1].y);
37. int i = 4;
38. for (int j = 1; j <= 3; j++) {
39. MoveToEx(hdc, p[i].x, p[i].y, NULL);
40. LineTo(hdc, p[i + j].x, p[i + j].y);}
```

Глобальна система координат.

```
41. p[8].x = 0.0;
42. p[8].y = 0.0;
43. p[8].z = 0.0;
44. p[9].x = 1.0;
45. p[9].y = 0.0;
46. p[9].z = 0.0;
47. p[10].x = 0.0;
48. p[10].y = 1.0;
49. p[10].z = 0.0;
50. p[11].x = 0.0;
51. p[11].y = 0.0;
52. p[11].z = 1.0;
```

```
53. for (int i = 1; i < 12; i++) {
```

Перехід з логічних в віконні координати.

```
54. p[i] = viewport.T(p[i]);}
55. i = 8;
56. for (int j = 1; j <= 3; j++) {
57. MoveToEx(hdc, p[i].x, p[i].y, NULL);
58. LineTo(hdc, p[i + j].x, p[i + j].y);}
59. int Engine::LoadBMP(HWND hWnd, HBITMAP hBmp, HDC hCmpDC, BITMAP bm,
    LPCWSTR fileName) {
```

Завантаження зображення.

```
60. hBmp = (HBITMAP)LoadImage(NULL, fileName, IMAGE_BITMAP, 0, 0,
    LR_LOADFROMFILE | LR_CREATEDIBSECTION);
61. if (hBmp == NULL){
62. MessageBoxW(hWnd, TEXT("Файл не найден"), TEXT("Загрузка изображения"),
    MB_OK | MB_ICONHAND);
63. DestroyWindow(hWnd);
64. return 1;}
65. GetObject(hBmp, sizeof(bm), &bm);
66. SelectObject(hCmpDC, hBmp);
67. ReleaseDC(hWnd, hCmpDC);
68. return 1;}
69. void Engine::Filter(RGBQUAD **rgb, RGBQUAD **rgb1, int mx, int my) {
70. int i, j;
71. double g[3][3];
```

Коефіцієнт нормування

```
72. double div = 1.0;
73. g[0][0] = 0.0; g[1][0] = 0.0; g[2][0] = 0.0;
74. g[0][1] = 0.0; g[1][1] = 1.0; g[2][1] = 0.0;
75. g[0][2] = 0.0; g[1][2] = 0.0; g[2][2] = 0.0;
76. RGBQUAD **rgbtmp = new RGBQUAD*[mx];
```

Усереднення (згладжування) кольорів пікселів відповідно до найближчими пікселями 1-го зображення.

```
77. for (i = 0; i < mx; i++) {
78. rgbtmp[i] = new RGBQUAD[my];}
79. for (j = 1; j < my - 1; j++) {
80. for (i = 1; i < mx - 1; i++) {
81. rgbtmp[i][j].rgbRed = (rgb[i - 1][j + 1].rgbRed*g[0][0] + rgb[i][j + 1].rgbRed*g[1][0] +
    rgb[i + 1][j + 1].rgbRed*g[2][0] + rgb[i - 1][j].rgbRed*g[0][1] + rgb[i][j].rgbRed*g[1][1]
    + rgb[i + 1][j].rgbRed*g[2][1] + rgb[i - 1][j - 1].rgbRed*g[0][2] + rgb[i][j -
    1].rgbRed*g[1][2] + rgb[i + 1][j - 1].rgbRed*g[2][2]) / div;
82. rgbtmp[i][j].rgbGreen = (rgb[i - 1][j + 1].rgbGreen*g[0][0] + rgb[i][j +
    1].rgbGreen*g[1][0] + rgb[i + 1][j + 1].rgbGreen*g[2][0] + rgb[i - 1][j].rgbGreen*g[0][1]
    + rgb[i][j].rgbGreen*g[1][1] + rgb[i + 1][j].rgbGreen*g[2][1] + rgb[i - 1][j -
    1].rgbGreen*g[0][2] + rgb[i][j - 1].rgbGreen*g[1][2] + rgb[i + 1][j -
    1].rgbGreen*g[2][2]) / div;
83. rgbtmp[i][j].rgbBlue = (rgb[i - 1][j + 1].rgbBlue*g[0][0] + rgb[i][j + 1].rgbBlue*g[1][0]
    + rgb[i + 1][j + 1].rgbBlue*g[2][0] + rgb[i - 1][j].rgbBlue*g[0][1] +
    rgb[i][j].rgbBlue*g[1][1] + rgb[i + 1][j].rgbBlue*g[2][1] + rgb[i - 1][j -
    1].rgbBlue*g[0][2] + rgb[i][j - 1].rgbBlue*g[1][2] + rgb[i + 1][j - 1].rgbBlue*g[2][2]) /
    div;}}
84. for (i = 1; i < mx - 1; i++) {
85. rgb[i] = rgbtmp[i];}
```

Усереднення (згладжування) кольорів пікселів відповідно до найближчими пікселями 2 - го зображення.

```
86. or (i = 0; i < mx; i++) {
87. rgbtmp[i] = new RGBQUAD[my];}
88. for (j = 1; j < my; j++) {
89. for (i = 1; i < mx; i++) {
90. rgb[i][j].rgbReserved = 128;
```

z = 0 rgb [i] [j] .rgbReserved використовується для зберігання координати Z (0-255).

```
91. rgb[i][j].rgbReserved = 128;}}
92. for (j = 1; j < my - 1; j++) {
93. for (i = 1; i < mx - 1; i++) {
94. rgbtmp[i][j].rgbRed = (rgb1[i - 1][j + 1].rgbRed*g[0][0] + rgb1[i][j + 1].rgbRed*g[1][0]
    + rgb1[i + 1][j + 1].rgbRed*g[2][0] + rgb1[i - 1][j].rgbRed*g[0][1] +
```

```
rgb1[i][j].rgbRed*g[1][1] + rgb1[i + 1][j].rgbRed*g[2][1] + rgb1[i - 1][j - 1].rgbRed*g[0][2] + rgb1[i][j - 1].rgbRed*g[1][2] + rgb1[i + 1][j - 1].rgbRed*g[2][2]) / div;
95. rgbtmp[i][j].rgbGreen = (rgb1[i - 1][j + 1].rgbGreen*g[0][0] + rgb1[i][j + 1].rgbGreen*g[1][0] + rgb1[i + 1][j + 1].rgbGreen*g[2][0] + rgb1[i - 1][j].rgbGreen*g[0][1] + rgb1[i][j].rgbGreen*g[1][1] + rgb1[i + 1][j].rgbGreen*g[2][1] + rgb1[i - 1][j - 1].rgbGreen*g[0][2] + rgb1[i][j - 1].rgbGreen*g[1][2] + rgb1[i + 1][j - 1].rgbGreen*g[2][2]) / div;
96. rgbtmp[i][j].rgbBlue = (rgb1[i - 1][j + 1].rgbBlue*g[0][0] + rgb1[i][j + 1].rgbBlue*g[1][0] + rgb1[i + 1][j + 1].rgbBlue*g[2][0] + rgb1[i - 1][j].rgbBlue*g[0][1] + rgb1[i][j].rgbBlue*g[1][1] + rgb1[i + 1][j].rgbBlue*g[2][1] + rgb1[i - 1][j - 1].rgbBlue*g[0][2] + rgb1[i][j - 1].rgbBlue*g[1][2] + rgb1[i + 1][j - 1].rgbBlue*g[2][2]) / div;}}
97. for (i = 1; i < mx - 1; i++) {
98. rgb1[i] = rgbtmp[i];}}
99. void Engine::GetZ(RGBQUAD **rgb, RGBQUAD **rgb1, int mx, int my) {
100.        _Point p;
101.        _Point p1;
102.        double al = 0.0, bt = 30.0, gm = 30.0;
103.        al = 3.14*al / 180;
104.        bt = 3.14*bt / 180;
105.        gm = 3.14*gm / 180;
106.        double dz = 0.01, pxy;
107.        int i, j, i1, j1, SUM, dmin;
108.        dmin = 80.0;
109.        int dR, dG, dB;
110.        for (j = 1; j < my; j++) {
111.        for (i = 1; i < mx; i++) {
112.        SUM = rgb[i][j].rgbRed + rgb[i][j].rgbGreen + rgb[i][j].rgbBlue;
```

Відфільтрувати світлі точки (255,255,255).

```
113.        if (SUM < 765)
```

Перехід від координат зображення (від низу до верху) до віконних координат (зверху вниз).

```
114.        {p.x = i;
115.        p.y = my - j;
116.        rgb[i][j].rgbReserved = 127;
117.        p.z = mx * rgb[i][j].rgbReserved / 255;
```

Виконується перетворення віконних координат в логічні координати x [-1, 1] y [-1, 1] z [-1, 1].

```
118.        p = viewport.T_inv(p);
```

Відфільтрувати точки за межами сфери одиничного радіуса.

```
119.        if (p.x*p.x + p.y*p.y + p.z*p.z < 1.0);
120.        do {p.z = p.z + dz;
```

Відфільтрувати точки за межами сфери одиничного радіуса.

```
121.        if (p.x*p.x + p.y*p.y + p.z*p.z >= 1.0) { p.z = 1.1; break; };
122.        p1.x = p.x*(cos(gm)*cos(al) - sin(gm)*sin(bt)*sin(al)) + p.y*cos(bt)*sin(al) -
            p.z*(sin(gm)*cos(al) + cos(gm)*sin(bt)*sin(al));
123.        p1.y = -p.x*(cos(gm)*sin(al) + sin(gm)*sin(bt)*cos(al)) + p.y*cos(bt)*cos(al) +
            p.z*(sin(gm)*sin(al) - cos(gm)*sin(bt)*cos(al));//p1.z = 0.0;
124.        p1 = viewport.T(p1);
```

Перехід від віконних координат до пікселів зображення (від низу до верху).

```
125.        i1 = p1.x;
126.        j1 = my - p1.y;
127.        dR = abs(rgb1[i1][j1].rgbRed - rgb[i][j].rgbRed);
128.        dG = abs(rgb1[i1][j1].rgbGreen - rgb[i][j].rgbGreen);
129.        dB = abs(rgb1[i1][j1].rgbBlue - rgb[i][j].rgbBlue);}
130.        while ((dR > dmin || dG > dmin || dB > dmin) && p.z <= 1.0);
131.        if (p.z <= 1) {
```

Перехід з логічних в віконні координати (головне зображення).

```
132.        p = viewport.T(p);
133.        rgb[i][j].rgbReserved = p.z * 255 / mx;
134.        else{
135.        rgb[i][j].rgbRed = 255; rgb[i][j].rgbGreen = 255; rgb[i][j].rgbBlue = 255;};}}}
136.        void Engine::ImagePixel(HDC hdc, RGBQUAD **rgb, int mx, int my) {
```

```
137.            COLORREF clr;
138.            _Point p;
139.            int i, j, SUM;
```

Виводимо результат.

```
140.        for (j = 0; j < my; j++) {
141.        for (i = 0; i < mx; i++) {
142.        SUM = rgb[i][j].rgbRed + rgb[i][j].rgbGreen + rgb[i][j].rgbBlue;
143.        if (SUM < 755) {
144.        p.x = i;
145.        p.y = my - j;
146.        p.z = mx * rgb[i][j].rgbReserved / 255;
147.        p = viewport.T_inv(p);
148.        action->CurrentMatrix.ApplyMatrixtoPoint(p);
149.        p = viewport.T(p);
```

Отрісовка точки.

```
150.        clr = RGB(rgb[i][j].rgbRed, rgb[i][j].rgbGreen, rgb[i][j].rgbBlue);
151.        SetPixel(hdc, p.x, p.y, clr);}}}}
152.        RGBQUAD ** Engine::MatrixBMP(const char *fname, int &mx, int &my)
153.        {FILE * pFile = fopen(fname, "rb");
```

Прочитуємо заголовок файлу.

```
154.        BITMAPFILEHEADER header;
155.        header.bfType = read_u16(pFile);
156.        header.bfSize = read_u32(pFile);
157.        header.bfReserved1 = read_u16(pFile);
158.        header.bfReserved2 = read_u16(pFile);
159.        header.bfOffBits = read_u32(pFile);
```

Прочитуємо заголовок зображення.

```
160.        BITMAPINFOHEADER bmiHeader;
161.        bmiHeader.biSize = read_u32(pFile);
162.        bmiHeader.biWidth = read_s32(pFile);
163.        bmiHeader.biHeight = read_s32(pFile);
164.        bmiHeader.biPlanes = read_u16(pFile);
```

```
165.        bmiHeader.biBitCount = read_u16(pFile);
166.        bmiHeader.biCompression = read_u32(pFile);
167.        bmiHeader.biSizeImage = read_u32(pFile);
168.        bmiHeader.biXPelsPerMeter = read_s32(pFile);
169.        bmiHeader.biYPelsPerMeter = read_s32(pFile);
170.        bmiHeader.biClrUsed = read_u32(pFile);
171.        bmiHeader.biClrImportant = read_u32(pFile);
172.        RGBQUAD **rgb = new RGBQUAD*[bmiHeader.biWidth];
173.        for (int i = 0; i < bmiHeader.biWidth; i++) {
174.        rgb[i] = new RGBQUAD[bmiHeader.biHeight];}
175.        int kr = (int)bmiHeader.biWidth * 3 % 4;
176.        if (kr != 0) {
177.        kr = 4 - kr;}
178.        for (int j = 0; j < bmiHeader.biHeight; j++) {
179.        for (int i = 0; i < bmiHeader.biWidth; i++) {
180.        rgb[i][j].rgbBlue = getc(pFile);
181.        rgb[i][j].rgbGreen = getc(pFile);
182.        rgb[i][j].rgbRed = getc(pFile);}
```

Пропускаємо останні 1-3 байта в кінці рядка для забезпечення кратності 4.

```
183.        for (int i = 0; i < kr; i++) {
184.        getc(pFile);}}
```

Передаємо значення ширини і висоти глобальних змінних.

```
185.        mx = bmiHeader.biWidth;
186.        my = bmiHeader.biHeight;
187.        fclose(pFile);
188.        return rgb;}
189.        unsigned short Engine::read_u16(FILE *fp)
190.        {unsigned char b0, b1;
191.        b0 = getc(fp);
192.        b1 = getc(fp);
193.        return ((b1 << 8) | b0);}
194.        unsigned int Engine::read_u32(FILE *fp)
195.        {unsigned char b0, b1, b2, b3;
```

```
196.        b0 = getc(fp);
197.        b1 = getc(fp);
198.        b2 = getc(fp);
199.        b3 = getc(fp);
200.        return ((((((b3 << 8) | b2) << 8) | b1) << 8) | b0);}
201.        int Engine::read_s32(FILE *fp)
202.        {unsigned char b0, b1, b2, b3;
203.        b0 = getc(fp);
204.        b1 = getc(fp);
205.        b2 = getc(fp);
206.        b3 = getc(fp);
207.        return ((int)(((((b3 << 8) | b2) << 8) | b1) << 8) | b0);}
```

Додаток В.

Перезентація магістерської дипломної роботи

Міністерство освіти і науки України
Східноукраїнський Національний Університету ім. В.Даля

ТЕХНОЛОГІЇ АНАЛІЗУ І ВІДНОВЛЕННЯ ТРИВМІРНОЇ ГРАФІЧНОЇ ІНФОРМАЦІЇ НА ОСНОВІ ДВОВИМІРНИХ ЗОБРАЖЕНЬ

Студент гр. КІ-18ДМ — Шаповалов О.О.
Керівник проекту — **Скарга-Бандурова І.С.**

1

Технічне завдання

Мета роботи

Метою роботи було підвищення точності побудови об'ємної моделі при незначному зростанні обчислювальної складності за рахунок розробки та впровадження алгоритма тривимірної реконструкцій із хмари точок на основі зіставлення двох зображень.

Етапи аналізу

- Аналіз сучасних підходів до тривимірної реконструкції, методів формування об'ємних зображень.
- Аналіз відомих алгоритмів обробки, визначення ключових характеристик методів: обчислювальна складність, точність побудови моделі.
- Оцінка ефективності методів.

2

Аналіз останніх публікацій тривимірної реконструкції

Дослідження	Завдання	Використовувані методи	Результати	Основні результати та висновки	Коментарі
Company C3D Labs, (2019)	Конвертація полігональної сітки в B-rep модель.	Сегментація полігональної моделі.	Програмний компонент C3D B-Shaper.	Метод трикутникової конвертації полігональної моделі.	У даній роботі пропонується метод і програмне забезпечення яке дозволить перетворювати полігональні моделі в граничні уявлення.
Michael Himpel; André Melzer (2019)	Three-Dimensional Reconstruction of Individual Particles in Dense Dust Clouds: Benchmarking Camera Orientations and Reconstruction Algorithms	This paper benchmarks two approaches capable of reconstructing the trajectories of particles in three dimensions. The influences of the particle number, the particle number density, and the orientation of the individual cameras are studied.	Three-dimensional algorithm for reconstructing individual particles in dense dust clouds	На основі методу мульти-тріангуляції	Даний алгоритм визначає тривимірне положення об'єкта в пиловій плазмі.
Фасхутдінов Р.Р. (2015)	Сегментація і реконструкція полігональної моделі по знімках комп'ютерної томографії нижніх кінцівок.	Воксельна модель; методи сегментації.	Програмний модуль автоматичної сегментації	Підхід комбінування методів.	В рамках даної роботи пропонується: програмний модуль для створення полігональної моделі по набору томографічних знімків.
Реут А.В. (2016)	Алгоритм візуалізації форми просторового об'єкта, поданого його матричною моделлю.	Воксельна модель.	Алгоритм візуалізації просторового тіла по його дискретної матричної моделі.	Побудова проміжної воксельної моделі поверхні тіла і визначення на її основі полігональної сітки з трикутними гранями.	В рамках даної роботи пропонується алгоритм який дозволяє пройти візуалізацію поверхні, що зменшує обсяг проміжних даних, усуваються неоднозначності скорочується процес обчислень.

3

Аналіз моделей, що використовуються в задачах тривимірної реконструкції

Моделі	Папери, де були використані ці моделі	Застосування моделей в задачах реконструкції	Плюси	Мінуси
Моделі на основі полігональних сіток	Company C3D Labs, (2019); Фасхітдінв Р.Р. (2015); Кулешов С.В. (2015); Гонахчян В.П. (2016).	Тривимірне сканування; моделювання за допомогою примітивів; реконструкція певного об'єкта зі сцени.	Висока точність моделі; набір інструментів побудови моделі; редагування моделі на етапі реконструкцій.	Використання технічного обладнання; високо обчислювальна складність; розпізнавання поверхонь різного типу.
Моделі типу грань-ребро-вершина;	Реут А.В. (2016); Гаранжі В.А., Кудрявцева Л.М (2012)	Математичне моделювання; побудова геометричних об'єктів.	Не висока складність обробки; висока обчислювальна швидкість реконструкції.	Втрата точності реконструкції об'єкта; реконструкція простих форм об'єкта.
Моделі на основі точкових уявлень (хмари точок);	Алсшбаев К.С. (2015); Аксьонов А.Ю. (2015) Юркін В.А. (2016). Шаповалов Р.В. (2010)	Моделювання ландшафту в ГІС системах; реконструкція об'єктів в руху, моделювання об'єктів з томографічних зображень; лазерне сканування.	Висока точність моделі; не висока складність обробки; розпізнавання тіла об'єкта.	Втрата обсягу структури тіла об'єкта; розриви між хмарами точок.
Воксельні моделі і моделі на основі октантних дерев;	Арсланов В.І. (2016); Вітіска Н.І. Гуляєв Н.А. (2016)	Реконструкція певного об'єкта зі сцени; математичне моделювання; побудова геометричних об'єктів.	Необмежений рівень деталізації; висока точність моделі.	Використання технічного обладнання; високо обчислювальна складність; статичність об'єкта; недолік оптимальних алгоритмів
Моделі, засновані на картах глибини	Краспльников Н.Н. (2015); Воронін В.В. (2014 року); Зачосом А. (2015).	Реконструкція кадрів відеоданих; Повна реконструкція сцени яка міститься в зображень	Реконструкція сцени; висока точність реконструкцій об'єктів на сцені; збереження розмірів об'єктів.	Недостатній рівень освітленості об'єкта; недостатній рівень надійності систем в зв'язку з можливістю надати неправдиві вхідні дані; велика залежність точності ідентифікації від положення об'єкта.

4

Оцінка і порівняння існуючих методів

Методи і алгоритми побудови об'ємної моделі

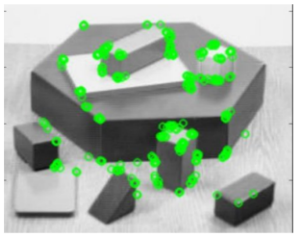
- Методи стерео зіставлення
- Методи обробки карти глибини
- Методи ректифікації
- Методи форми від фокусу
- Алгоритм реконструкцій із хмари точок

Назва методу	Обчислювальна складність	Точність
Локальний метод пошуку.	O(WHD)	73%
Динамічне програмування для локального методу пошуку.	O(WH)	85%
Динамічне програмування для глобального методу пошуку.	$O((WH)^3 \log(WH))$	86%
Алгоритми на основі випадкових полів Маркова. Алгоритм розрізу графа.	$2^{(\log(a))}$	92%
Алгоритми на основі випадкових полів Маркова. Алгоритм поширення довіри.	$O(WHD^2T)$	79%
Полуглобальний алгоритм	O(WHD)	88%

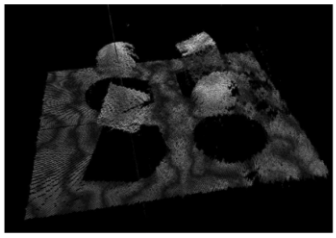
5

Оцінка комбінування методів

Для досягнення поставленої задачі, методи та алгоритми можуть бути скомбіновані для підвищення точність побудови тривимірної моделі. Такими прикладом комбінування є алгоритми: Oriented FAST and Rotated BRIEF (ORB); Гібридний паралельний алгоритм зіставлення стереозображення; Локальний метод пошуку.



Oriented FAST and Rotated BRIEF (ORB)



Гібридний паралельний алгоритм зіставлення стереозображення

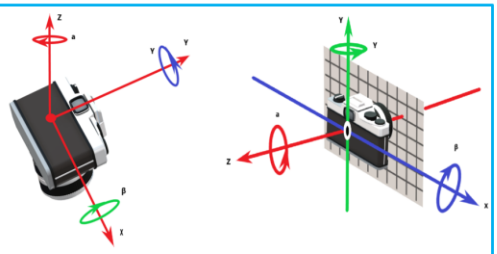
Локальний метод пошуку

6

Постановка завдання і рішення

Суть математичного рішення задачі відновлення тривимірного об'єкту за двома або більше зображень полягає в описі зміни положення датчиків (камери) захоплення проекції при виконанні різних кадрів. Положення визначається переносом і обертанням.

Осі системи координат (СК) пристрої виходять з центру пристрій (камеру). Обертання визначається кутами Ейлера (alpha beta gamma), які представляють собою різницю в градусах між координатами пристрою і земними координатами.



Орієнтація пристрою щодо координат

7

Схема визначення математичної залежності між координатами точок 2-х зображень

Точки описані в площині *XY* (*xy*) головного та допоміжного зображень, вісь *Z* (*z*) спрямована перпендикулярно до площини камери і проходить через її центр. Головним є те зображення, фото якого отримано першим.

Значення кутів визначається різницею (дельта) кутів допоміжного положення камери щодо головного:

$$\triangle a = a_1 - a_0$$

$$\triangle \beta = \beta_1 - \beta_0$$

$$\triangle Y = Y_1 - Y_0$$

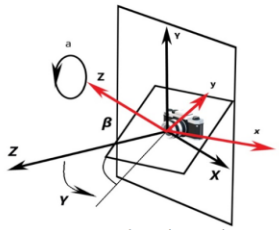


Схема взаємозв'язків орієнтації двовимірного зображення

8

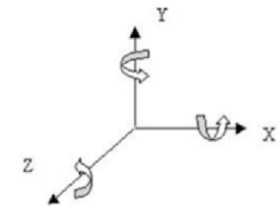
Визначення відповідності між координатами точок зображень

Положення допоміжної СК *xyz* щодо головної *XYZ* задається композицією 3-х матричних перетворень - обертання щодо осі *Z*, обертання щодо осі *X* і обертання щодо осі *Y*.

$$R_z R_x R_y$$

$$R_z = \begin{pmatrix} \cos a & \sin a & 0 & 0 \\ -\sin a & \cos a & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_y = \begin{pmatrix} \cos Y & 0 & -\sin Y & 0 \\ 0 & 1 & 0 & 0 \\ \sin Y & 0 & \cos Y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Композиція 3-матричних перетворень, обертання щодо осі Z

9

Визначення відповідності між координатами точок зображень

Для перерахунку точок об'єкта з ГСК *XYZ* в ДСК *xyz* необхідно отримати результат перемноження зворотних матриць в послідовності навпаки. Результат множення трьох зворотних матриць і матриці ортогонального перетворення дає наступний результат:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} T_p^{-1} = \begin{pmatrix} \cos Y \cos \alpha - \sin Y \sin \beta \sin \alpha & -\cos Y \sin \alpha - \sin Y \sin \beta \cos \alpha & 0 & 0 \\ \cos \beta \sin \alpha & \cos \beta \cos \alpha & 0 & 0 \\ -\sin Y \cos \alpha - \cos Y \sin \beta \sin \alpha & \sin Y \sin \alpha - \cos Y \sin \beta \cos \alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Перетворення координат точок з головною СК у допоміжну СК в матричному поданні має вигляд:

$$(xyz1) = (XYZ1)T_p^{-1}$$

У підсумку, рівняння перетворення точок просторового об'єкта в точки поточного зображення запишуться у вигляді:

$$x = X(\cos Y \cos \alpha - \sin Y \sin \beta \sin \alpha) + Y \cos \beta \sin \alpha - Z(\sin Y \cos \alpha + \cos Y \sin \beta \sin \alpha)$$

$$y = -X(\cos Y \sin \alpha + \sin Y \sin \beta \cos \alpha) + Y \cos \beta \cos \alpha + Z(\sin Y \sin \alpha - \cos Y \sin \beta \sin \alpha)$$

$$Z = 0$$

10

Опис кроків роботи алгоритму

Система рівнянь дозволяє знайти координату *Z* кожної точки *(X, Y)* на головному зображенні, якщо відомі кути і відповідне місце на *(x, y)* на допоміжному зображенні. Відповідність між точками може встановлюватися за їх кольором.

1) Для кожної точки головного зображення *(X, Y)* визначаємо відповідну точку на допоміжному зображенні *(x, y)*, здійснюючи ітерацію координати *Z*.

2) Відповідна точка знайдена, якщо збігаються 3 кольору точок (RGB) на обох зображеннях.

3) Разом з відповідною точкою визначається і координата *Z*.

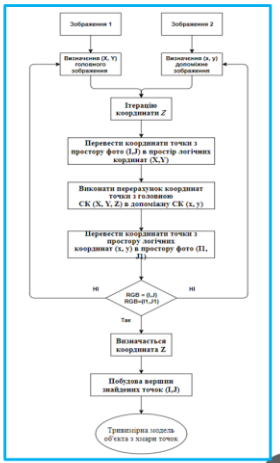
Для пошуку потрібної точки на двох зображеннях необхідно в ітераційному режимі:

1) перевести координати точки з простору фото *(I, J)* в простір логічних координат *(XY)* задати координату *Z (Z = Z + dZ)*;

2) виконати перерахунок координат точки з головною СК *(X, Y, Z)* в допоміжну СК *(x, y)*;

3) перевести координати точки з простору логічних координат *(x, y)* в простору фото *(I1, J1)*;

4) перевірити на відповідність за кольором точки головного зображення *(I, J)* і допоміжного *(I1, J1)*.



11

Тестування алгоритму

На основі запропонованого алгоритму було розроблено програмне забезпечення тривимірної реконструкцій на мові C++. Для тесту алгоритму було підготовлено набір вхідних зображень отриманих з бібліотеки тривимірних об'єктів Paint 3D. У редакторі тривимірної моделі встановлювалися параметри об'єкта щодо осі координат *alpha (Z), beta (X) і gamma (Y)* шляхом зміни положення об'єкта в тривимірному просторі. Отримане положення об'єкта конвертується в двомірне зображення.

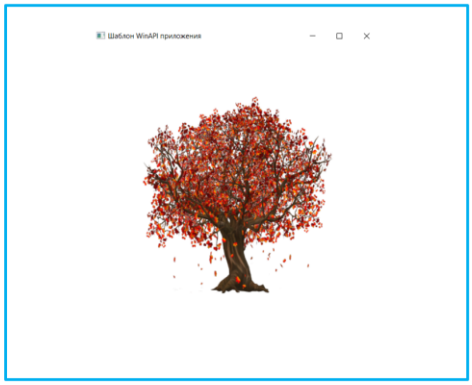
Номер рисунку	Alpha	Beta	Gamma	Інтерацііїя	Колірної параметр
Тест 1	00.00	30.00	30.00	0.01	80.00
Тест 2	00.00	-32.00	20	0.01	70.00
Тест 3	00.00	70	50	0.01	30.00
Тест 4	10.00	30.00	10.00	0.01	60

Параметри вхідних зображень

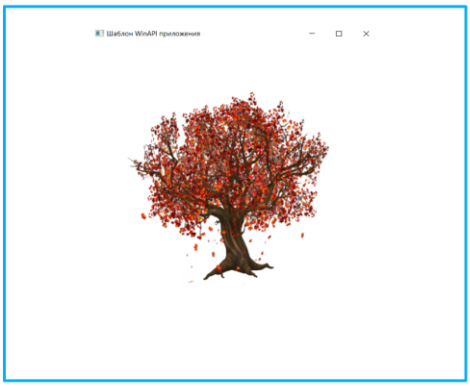
12

Тестування алгоритму

1) Завантаження головного тестового зображення.



2) Завантаження допоміжного тестового зображення.

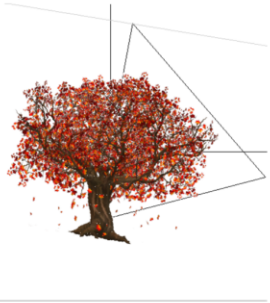


13

Тестування алгоритму

3) Перевірка на відповідність за кольором точки головного тестового зображення і допоміжного тестового зображення.

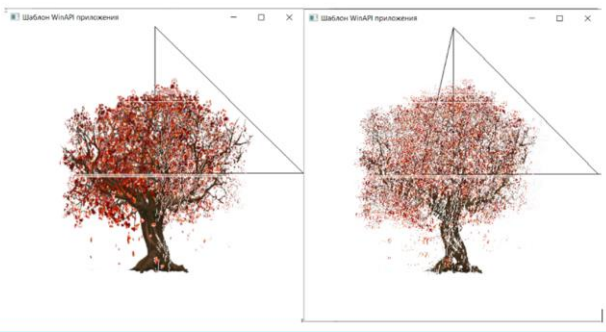
4) Визначення ітерації по координаті Z = 0.



14

Тестування алгоритму

5) Побудова вершин точок по знайденим відповідностям. Проектування хмари точок в тривимірному просторі.



15

Аналіз запропонованого алгоритму

Обчислювальна складність алгоритму є O(WHN), де W - ширина зображення, H - висота, а N - кількість відповідностей кольорових точек на головному і допоміжному зображенні.

Номер тесту	Обчислювальна складність	Точність	Час роботи в мікросекундах (CPU)			
			Відповідність за кольором	Ітерація	Проектування хмари точок	Загальна сума часу
Тест 1	O(WHN)	92.73%	0.234	0.283	0.632	1.15
Тест 2	O(WHN)	91.26%	0.241	0.27	0.65	1.161
Тест 3	O(WHN)	89.53%	0.273	0.29	0.67	1.233
Тест 4	O(WHN)	86.08%	0.28	0.312	0.693	1.29

Виходячи з аналізу проведених тестів в середньому точність побудованої тривимірної моделі близько 90%. В середньому швидкість виконання алгоритму становить 1 секунда.

16

Порівняльний аналіз алгоритмів

Для проведення порівняльного аналізу алгоритму були обрані існуючі метод і алгоритм.

Назва методу	Обчислювальна складність	Точність
Запропонований алгоритм	O(WHN)	90%
Локальний метод пошуку.	O(WHD)	73%
Евристичний алгоритм сегментації хмари точок	$O(n * logn * \frac{2\pi}{k} + \frac{n}{\varepsilon^3})$	90%

Виходячи з порівняльного аналізу можна зробити висновок що метод локального пошуку має невисоку обчислювальну складність, але показує в 74% відсотка точність побудови моделі. Евристичний алгоритм сегментації хмари точок показує хорошу точність відновлення моделі але має високу обчислювальну складність, оскільки алгоритм зіставляє вершину графа з кожним отриманим сегментом хмари точок.

17

ВИСНОВОК

В ході магістерської роботи були досягнуті такі результати як: проведена класифікація методів тривимірної реконструкції; аналіз останніх публікацій тривимірної реконструкції та аналіз моделей, що використовуються в задачах тривимірної реконструкції; проведено аналіз сучасних алгоритмів і методів тривимірної реконструкцій; проведена оцінка і порівняння методів; проведена оцінка комбінування методів.

У ході практичної частини роботи були отримані наступні результати: розроблено алгоритм тривимірної реконструкції з хмари точок на основі методу стерео зіставлення; на основі запропонованого алгоритму було розроблено програмне забезпечення тривимірної реконструкцій на мові C++; проведено тестування алгоритму; проведено аналіз запропонованого алгоритму; проведено порівняльний аналіз алгоритмів.

18