МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Визначення координати положення БПЛА за картинкою / відео за допомогою афінних перетворень»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Довбиш А.С.**

Студента групи ІН.м - 92 — **Охріменко К.П.**

СУМИ 2020

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *«Інформатика»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Охріменку Кирилу Павловичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Визначення координати положення БПЛА за картинкою / відео за допомогою афінних перетворень*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми. Постановка задачі дослідження. 2) Огляд існуючих методів детектування та поєднання їх з попередньою роботою. 3) Використання Афінних перетворень для визначення змін у положенні апарату. 4) Розробка інформаційного та програмного забезпечення системи визначення координати положення БПЛА.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) ___
_____
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми. Постановка задачі дослідження*		
2.	*Огляд існуючих методів детектування та поєднання їх з попередньою роботою*		
3.	*Використання Афінних перетворень для визначення змін у положенні апарату*		
4.	*Розробка інформаційного та програмного забезпечення системи визначення координати положення БПЛА*		
5.	*Оформлення пояснювальної записки до дипломної роботи*		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 26 стор., 10 рис., 1 додаток, 11 джерел.

Об’єкт дослідження — система визначення координати положення БПЛА за допомогою афінних перетворень.

Мета роботи — розробка алгоритму для розпізнавання координат БПЛА за рахунок пошук змін у координатах об’єкту-якорю, а саме за допомогою визначення афінної матриці перетворення у різні моменти часу.

Методи дослідження — метод знаходження афінного перетворення по точках.

Результати — проведено огляд літературі та створеного програмного забезпечення, на основі якого розроблено алгоритм пошуку афінної матриці перетворення у тривимірному просторі, проаналізовано графічні зміни знімків об’єкту з камери апарату. Після аналізу матриці перетворення було визначено координати об’єкту. Дані методи запрограмовані на платформі MATLAB.

MATLAB, SURF ДЕСКРИПТОРИ, ДЕТЕКТУВАННЯ КЛЮЧОВИХ ТОЧОК ЗОБРАЖЕННЯ, АФІННА СИСТЕМА КООРДИНАТ, АФІННІ ПЕРЕТВОРЕННЯ

ЗМІСТ

ВСТУП

Сьогодні вже нікого не можна здивувати тим, що якусь задачу виконують автономні механізми: починаючи з роботів прибиральників та закінчуючи роботами мінерами. Проте, мабуть найпоширенішими серед них є квадрокоптери, і це не дивно – спектр задач, що може виконувати апарат, доволі різноманітний. Перевозка товарів, пошук об’єктів на місцевості, аналіз навколишнього середовища, тощо – всі ці задачі потребують не тільки самого БПЛА, а й аналізу отриманих даних самим користувачем. Тому все частіше постає питання про створення автономного апарату, котрий мінімально залежав би від постійного контролю оператором. Від цієї «залежності» можна звільнитися різними шляхами – намагатися передбачити усі можливі ситуації під час роботи, максимально зменшити вхідні данні від оператора, отримуючи від нього лише основні положення для подальших дій, або ж навпаки розробити алгоритм, що сам аналізує отримані данні та на їх основі самостійно приймає рішення.

Основним завданням цієї роботи буде розробка алгоритму, що дозволить визначити координати положення БПЛА на основі даних, котрі було отримано з камер, встановлених у апарат, та подальшої передачі даних оператору. У бакалаврській роботі було розглянуто методи для ідентифікації об’єкту на знімку та створено алгоритм, котрий може знаходити об’єкт навіть при зміні ракурсу чи відстані. Саме ці зміни й будуть вхідними даними для поточної задачі.

Далі буде розглянуто найпоширеніші способи та методи знаходження перетворення по точкам, рішення системи лінійних рівнянь, перетворення координат при зміні базису і тому подібних перетворень плоскостей об’єкту. Буде обрано та програмно реалізовано алгоритм, що зможе визначені зміни в положенні БПЛА на основі зазначених вище перетворень.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Вхідні дані

Завдання цієї роботи - визначення координати положення БПЛА за картинкою чи відео за допомогою афінних перетворень – не являється відокремленою задачею, а скоріше набором кількох різних менших задач. Хоча ці задачі і відносяться до різних наукових сфер (наприклад ідентифікація об’єкту на знімку, тріангуляція та перетворення системи координат), нас цікавить їх взаємозв’язок та можливість перетворення результатів однієї у вхідні данні для іншої.

Оскільки ця робота є продовженням бакалаврської, будуть використані минулі наробітки, а саме:

- Ідентифікація об’єкту за допомогою методу SURF (Speeded Up Robust Features)
- Частина алгоритму візуальної одометрії

SURF (Speeded Up Robust Features).

Метод SURF вирішує два завдання - пошук особливих точок зображення і створення їх дескрипторів (описового елемента, інваріантного до зміни масштабу і повороту). Крім того, сам пошук ключових точок теж повинен володіти инвариантностью, тобто повернений об'єкт сцени повинен володіти тим же набором ключових точок, що і зразок. Метод шукає особливі точки за допомогою матриці Гессе. Після знаходження ключових точок, метод SURF формує їх дескриптори.

В цілому, метод справляється із завданням розпізнавання образів васкулярного малюнка долоні. Чим менше ділянка, тим менше на нього впливають великомасштабні спотворення. Так, якщо об'єкт в цілому, піддається ефекту перспективи (ближній край об'єкта має більший видимий розмір, ніж

далекий), для малого його ділянки явищем перспективи можна знехтувати, замінивши масштабуванням.

Рис. 1.1 - Приклад знаходження ключових точок для частино скритого об’єкту та дещо зміненого ракурсу

Аналогічно, невеликий поворот об'єкту навколо осі може сильно змінити картинку об'єкта в цілому, однак малі ділянки зміняться незначно. Крім того, в разі, коли частина об'єкта виходить за край зображення або закрита, невеликі ділянки навколо частини ключових точок видно цілком, що також дозволяє їх ідентифікувати

ВІЗУАЛЬНА ОДОМЕТРІЯ.

Візуальна одометрія - метод оцінки стану і орієнтації об'єкта за допомогою аналізу послідовності зображень знятих встановленою на ньому камерою (камерами). Завдяки цьому методу можна отримати інформацію про напрямок руху та пройдену дистанцію. Візуальна одометрія дозволяє побудувати систему навігації для апарату будь-якого типу пересування та на будь-якій поверхні.

В залежності від кількості використовуємих відеокамер, дана задача може мати різні розв’язки та методи, що використовуються для цього. У нашому випадку необхідно знайти рішення задачі monocular odometry, тобто разробити

алгоритм візуальної одометрії, що використовує данні від однієї відеокамери. Проте він не буде значною мірою відрізнятись від стандартного алгоритму.

Стандартний алгоритм візуальної одометрії:

1. Отримання зображення з камери
2. Корекція зображення
3. Детектування ключових точок зображення
4. Перевірка векторів оптичного потоку на потенціальні помилки
5. Визначення руху камери з оптичного потоку
6. Оновлення ключових точок

Оптичний потік - це зображення видимого руху об'єктів, поверхонь або країв сцени, що отримується в результаті переміщення спостерігача (камери) щодо сцени.

Безспірною перевагою даного методу являється його універсальність. До мінусів можна віднести наступне:

- Погана робота алгоритму на однотипних зображеннях
- Необхідність високої швидкості захвату зображення
- Висока розрахункова загрузка

1.2 Аналіз проблеми

Отже, маючі в якості вхідних даних фото об’єкту під різним кутом, на різній відстані або ж з певним зміщенням, перед нами постає задача в знаходженні певних ключових точок об’єкту та їх взаємному розташуванні на різних знімках. Ще під час минулої роботи було вирішено, що у якості об’єкту будуть виступати фанфарні стовпи або ж ЛЕП, оскільки вони виділяються на місцевості та встановлюються на певній відстані один від одного згідно нормам, тобто вони є більш передбачуваними орієнтирами. Проте це не значить, що вони єдиний можливий варіант – якщо на місцевості можливо знайти орієнтири з подібними характеристиками (наприклад багатоповерхові будівлі), програму можна буде скорегувати на пошук нового орієнтиру.

Головною проблемою в цьому випадку є те, що отримані знімки являються двовимірними, тобто визначити як сама змінилося положення БПЛА лише по ним – доволі важка задача. Необхідно ввести у апарат тривимірну карту місцевості, на котру надалі вже накладатимуться отримані фото. Або ж передавати дані з апарату користувачеві, котрий аналізуватиме їх у порівнянні з місцевістю, що підвищить точність показників.

Оскільки нас цікавить рух у тривимірному просторі, то й точки характеризуватимуться трьома координатами (x, y, z) та вектором напрямку, куди саме змістилась точка на наступному кадрі. Головну увагу будемо приділяти афінній системі координат в просторі.

Сукупність точки і базису називається декартовою (або афінною) системою координат:

- аффинная система координат на прямій (рис.1.2,а) - це точка і ненульовий вектор на прямий (базис на прямій);
- аффинная система координат на площині (рис.1.2,б) - це точка і два неколінеарних вектора, взяті в певному порядку (базис на площині);
- аффинная система координат в просторі (рис.1.2,в) - це точка і три некомпланарних вектора, взяті в певному порядку (базис в просторі).

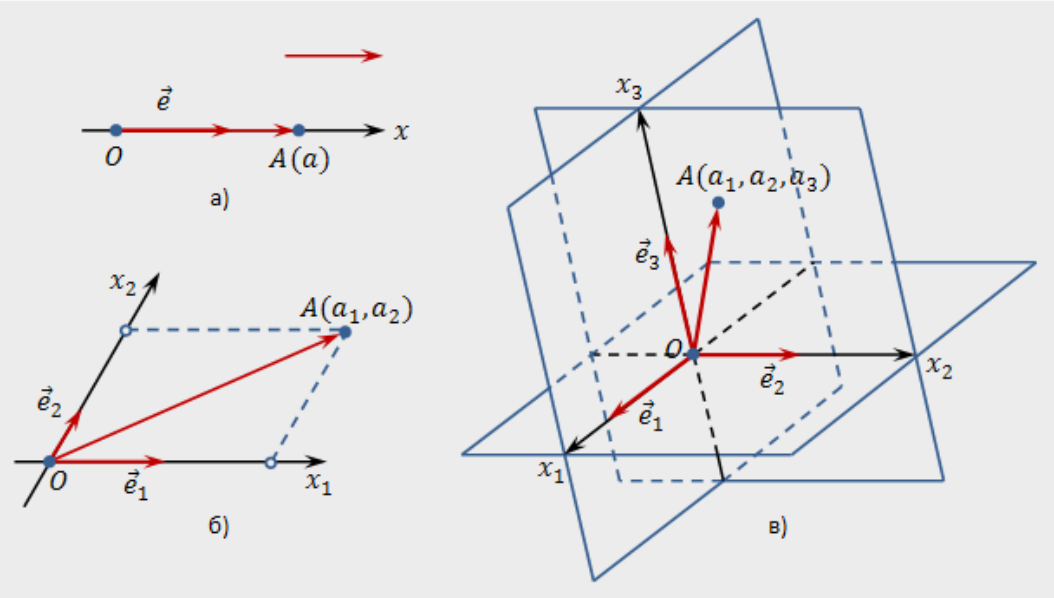

Рис. 1.2 - Приклад знаходження ключових точок для частино скритого об’єкту та дещо зміненого ракурсу

Надалі необхідно буде знайти формули та методи, що зв’язували б старі та нові координати точок. Існує декілька варіантів вирішення цього завдання, проте в нашому випадку будуть використані саме Афінні перетворення. Їх переваги та недоліки буде розглянуто в наступних розділах цієї роботи.

Серед цих перетворень нас цікавить матриця переходу від старого базису до нового (матрицею перетворення базису). Користуючись її властивостями, буде вирахувано пройдену відстань.

Наступним кроком буде реалізувати алгоритм пошуку матриці переходу програмними засобами та інтегрувати результати бакалаврської роботи у якості вхідних даних для цього методу. В якості вихідних даних буде отримано направляючий вектор, котрий допоможе визначити та обчислити пройдений апаратом маршрут.

1.3 Постановка задачі

Програмне забезпечення створюється для БПЛА, в пам’ять котрого буде внесено шаблон об’єкту, на основі якого надалі відбуватиметься пошук орієнтиру на місцевості. Під час польоту будуть робитися знімки місцевості, на яких буде ідентифіковано шуканій об’єкт, і за змінами ключових точок об’єкту будо визначено положення апарату.

Таким чином метою роботи є:

1) створення алгоритму для встановлення перетворення координат в просторі;

2) переведення результатів бакалаврської роботи у вхідні данні для попереднього пункту;

3) визначення шляху, пройденого апаратом за даними з попереднього пункту.

2 МЕТОДИ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 Причини обрання Афінних перетворень

Оскільки працюємо з БПЛА, ми будемо використовувати тривимірну систему координат, тобто основні операції будуть відбуватися з точками та їх координатами, а також вектором–показчиком зміни точки у просторі.
Було обрано саме Афінні перетворення, оскільки вхідні і вихідні вектори можуть мати різну розмірність. Формули, застосовна для афінних перетворень, діють у просторах будь-якої розмірності. Втім, вхідних точок повинно бути достатньо і вони не повинні «злипатися»: якщо Афінні перетворення діють з тримірного простору - точки повинні утворювати невироджених симплекс з 4 точок. Якщо ця умова не виконана, то однозначно відновити перетворення неможливо (ніяким методом взагалі, не тільки цим) - формула попередить про це нулем в знаменнику.

2.2 Система Афінних координат

Спочатку необхідно вирішити, як саме буде задоно Афінну систему координат, що відповідатиме поняттю точка, вектор, тощо.

Координати точки записують у вигляді A (a_1, a_2, a_3)

Точка O називається початком координат. Прямі, що проходять через початок координат в напрямку базисних векторів, називаються координатними осями: Ox_1- вісь абсцис, Ox_2- вісь ординат, Ox_3- вісь аплікат. Площині, що проходять через дві координатні осі, називаються координатними площинами.
Для будь-якої точки A в заданій афінній системі координат можна розглянути вектор $\vec{OA}$ початок якого збігається з початком координат, а кінець - з точкою A. Цей вектор називається радіус-вектором точки A.

Координатами точки A в заданій системі координат називаються координати радіус-вектора цієї точки відносно заданого базису. У просторі це

координати вектора $\vec{OA}$ в базисі $\vec{e}_1, \vec{e}_2, \vec{e}_3$ тобто коефіцієнти a_1, a_2, a_3 в розкладанні $\vec{OA} = a_1 * \vec{e}_1 + a_2 * \vec{e}_2 + a_3 * \vec{e}_3$.

2.3 Аффінні перетворення

Спочатку розглянемо основи для двувимірного простору.

Нехай в просторі задані дві системи координат - $O\vec{e}_1, \vec{e}_2, \vec{e}_3$ та $O\vec{e}'_1, \vec{e}'_2, \vec{e}'_3$. Перша буде відповідати першому знімку, а друга – наступному.

Тоді базис $\vec{e}_1, \vec{e}_2, \vec{e}_3$ (старий) і базис $\vec{e}'_1, \vec{e}'_2, \vec{e}'_3$, (новий) представимо у вигляді матриць-рядків $(\vec{e}) = (\vec{e}_1\ \vec{e}_2\ \vec{e}_3)$ та $(\vec{e}') = (\vec{e}'_1\ \vec{e}'_2\ \vec{e}'_3)$. Нас цікавить задача знаходження аффінного перетворення за точками з подальшим знаходженням пройденого шляху.

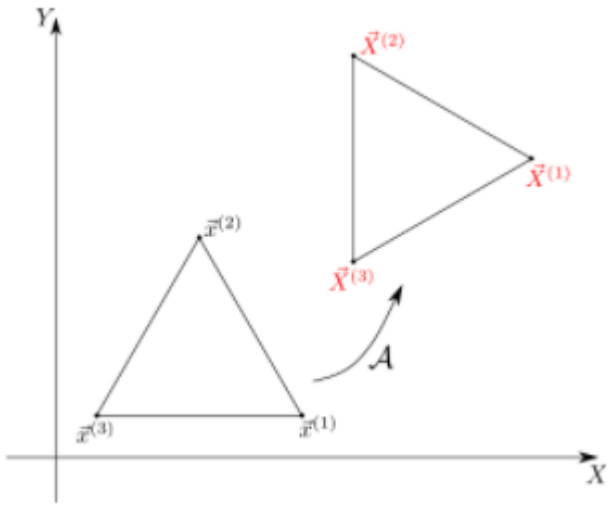

Рис. 2.1 – Приклад Афінного перетворення $\mathcal{A}$

Для цього необхідно буде знайти координати векторів нового базису відносно старого базису і записати їх у вигляді

$$\begin{cases} \vec{e}'_1 = s_{11} \cdot \vec{e}_1 + s_{21} \cdot \vec{e}_2 + s_{31} \cdot \vec{e}_3, \\ \vec{e}'_2 = s_{12} \cdot \vec{e}_1 + s_{22} \cdot \vec{e}_2 + s_{32} \cdot \vec{e}_3, \\ \vec{e}'_3 = s_{13} \cdot \vec{e}_1 + s_{23} \cdot \vec{e}_2 + s_{33} \cdot \vec{e}_3. \end{cases}$$

Записуючи за стовпцями координати векторів $\vec{e_1'}, \vec{e_2'}, \vec{e_3'}$, щодо $\vec{e_1}, \vec{e_2}, \vec{e_3}$, складаємо матрицю

$$\underset{(\vec{e})\to(\vec{e}\,')}{S} = \begin{pmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \\ s_{31} & s_{32} & s_{33} \end{pmatrix}$$

Квадратна матриця $\underset{(\vec{e})\to(\vec{e^1})}{S}$, складена з координатних стовпців векторів нового базису $(\vec{e^1})$ в старому базисі $(\vec{e})$. Дана матриця і буде матрицею перетворення базису.

Розглянемо застосування афінних перетворень для тривимірного простору. Для початку введемо поняття "однорідних координат".

Замінимо координатну трійку (x, y, z), задану точку в просторі, на четвірку чисел (x, y, z, 1) або, більш загально, на четвірку $(hx, hy, hz, h), h \neq 0.$

Кожна точка простору (крім початкової точки O) може бути задана четвіркою водночас не рівних нулю чисел; ця четвірка чисел визначена однозначно з точністю до загального множника.

Запропонований перехід до нового способу завдання точок дає можливість скористатися матричним записом і в більш складних, тривимірних задачах.

Будь-яке афінне перетворення в тривимірному просторі може бути представлене у вигляді суперпозиції обертань, розтягнень, віддзеркалень і переносів. Тому цілком доречно спочатку детально описати матриці саме цих перетворень.

- **Матриці обертання в просторі.**

Матриця обертання навколо осі абсцис на кут φ:

$$[R_x]=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\varphi & \sin\varphi & 0\\ 0 & -\sin\varphi & \cos\varphi & ?\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Матриця обертання навколо осі ординат на кут ψ:

$$[R_y]=\begin{bmatrix}\cos\psi & 0 & -\sin\psi & 0\\ 0 & 1 & 0 & 0\\ \sin\psi & 0 & \cos\psi & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Матриця обертання навколо осі аплікат на кут χ:

$$[R_y]=\begin{bmatrix}\cos\chi & \sin\chi & 0 & 0\\ -\sin\chi & \cos\chi & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

- **Матриця розтягування (стиснення).**

$$[D]=\begin{bmatrix}\alpha & 0 & 0 & 0\\ 0 & \beta & 0 & 0\\ 0 & 0 & \gamma & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.$$

α> 0 - коефіцієнт розтягування (стискання) уздовж осі абсцис

β> 0 - коефіцієнт розтягування (стискання) уздовж осі ординат

γ> 0 - коефіцієнт розтягування (стискання) уздовж осі аплікат.

- **Матриці відображення.**

Матриця відображення відносно площини Xy:

$$[M_z] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Матриця відображення відносно площини Yz:

$$[M_x] = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Матриця відображення відносно площини Zx:

$$[M_y] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Матриця переносу (тут (λ, μ, ν) - вектор переносу).**

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \lambda & \mu & \nu & 1 \end{bmatrix}.$$

Таким чином, будь-яке складне аффінне перетворення може бути досить просто отримано шляхом комбінації чотирьох базових перетворень, а пройдений шлях може бути визначений за допомогою формули для визначення відстані між точками в просторі:

$$|AB| = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + \left(z_B - z_A\right)^2}$$

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Опис обраного ПО

Описані вище перетворення використовуються в багатьох програмних продуктах на різноманітних мовах програмування, проте зазвичай вони «заточені» для виконання специфічних задач, тобто можуть допускати похибки у процесі обчислення або одразу перетворювати необхідний нам результат у щось інше. Однак в MATLAB ці функції реалізовані доволі чітко, до того ж користувач може отримати до них доступ у будь-який момент та скоректувати згідно з поставленою задачею чи новими вхідними даними. MATLAB написаний на мові C\C++, а отже може бути перенесений на різні операційні системи, зокрема на Linux, що може пришвидшити обробку даних та виконання алгоритмів.

Проте, оскільки MATLAB є комерційним продуктом, для повсякчасного використання розробленої системи необхідно перейти на ПО, що має відкритий код. У цьому випадку можна використати систему для математичних розрахунків GNU Octave, оскільки вона використовує сумісний з MATLAB язик програмування.

При роботі з MATLAB було використано наступні встроєні бібліотеки:

- pctransform
- estimateGeometricTransform3D
- imshowpair
- transformPointsForward

Важливо зазначити, що для пошуку ключових точок MATLAB необхідно перевести зображення у сіру гамму.

3.2 Програмна реалізація

```
%{ read object and scene }%
I1 = imread('pik1.jpg');
objImage1 = rgb2gray(I1);
dicomwrite(objImage1, 'pik1.dcm');
fixed = dicomread('pik1.dcm');

I2 = imread('pik2.jpg');
objImage2 = rgb2gray(I2);
dicomwrite(objImage2, 'pik2.dcm');
moving = dicomread('pik2.dcm');

figure;
subplot(1,2,1);
imshow(fixed);
hold on;
subplot(1,2,2);
imshow(moving);
hold on;
imshowpair(fixed, moving,'Scaling','joint')
suptitle('.jpg to .dcm');
```

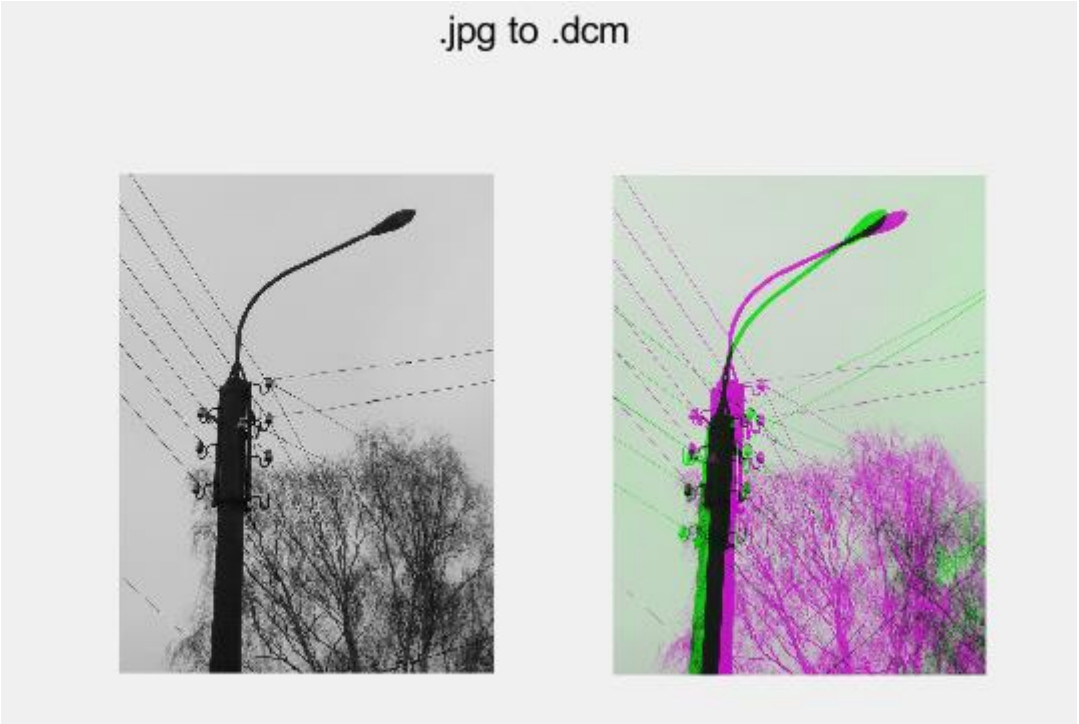

Рис. 4.1 - Зчитані зображення, переведені в сіру гамму (.dcm файли) з додатковим порівнянням одного й того ж об’єкту з різних ракурсів

```
%{ generate first object and move it for some ditance }%
[theta, r, h] = meshgrid(0:.1:6.28, 0.5, 0:.2:8); % making a cylinder
r = r + 0.05 * randn(size(r)); % adding some radial noise
[x, y, z] = pol2cart(theta, r, h); % transforming the coordinate system
P1 = ([x(:), y(:), z(:)]')'; % rotating the data points around x axis
P2 = (rotx(60) * [x(:), y(:), z(:)]')'; % rotating the data points around x axis
figure;
scatter3(P1(:, 1), P1(:, 2), P1(:, 3))
scatter3(P2(:, 1), P2(:, 2), P2(:, 3))
axis equal
```

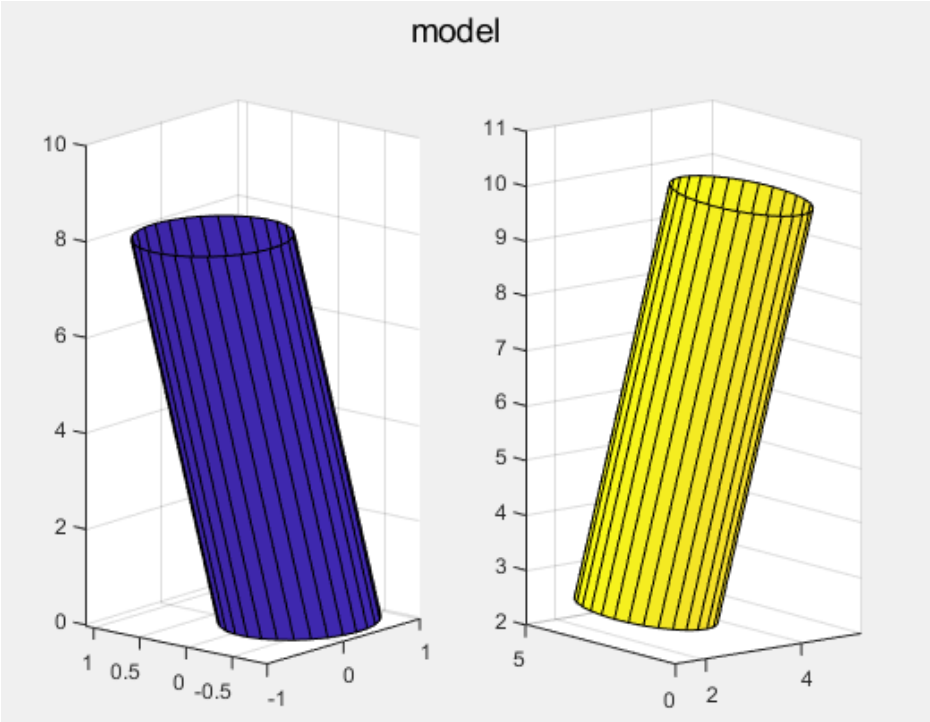

Рис. 4.2 – Змодельовані геометричні об’єкти, з якими надалі будуть відбуватися перетворення

```
pC1 = pointCloud(P1);
pC2 = pointCloud(P2);

%{ create some generic transformation }%
theta = 30;
rot = [cosd(theta)  sind(theta) 0; ...
      -sind(theta)  cosd(theta) 0; ...
            0           0       1];
trans = [4 3 2];
tform = rigid3d(rot,trans);

figure
pcwrite(pC1,'ptCloud1.ply','Encoding','ascii');
ptCloud1 = pcread('ptCloud1.ply');
ptCloud1 = pcdownsample(ptCloud1,'random',0.25);
pcshow(ptCloud1);
title('Point Cloud of a first object')

figure
ptCloud2 = pctransform(ptCloud1,tform);
pcshow(ptCloud2);
title('Point Cloud of a second object')
```

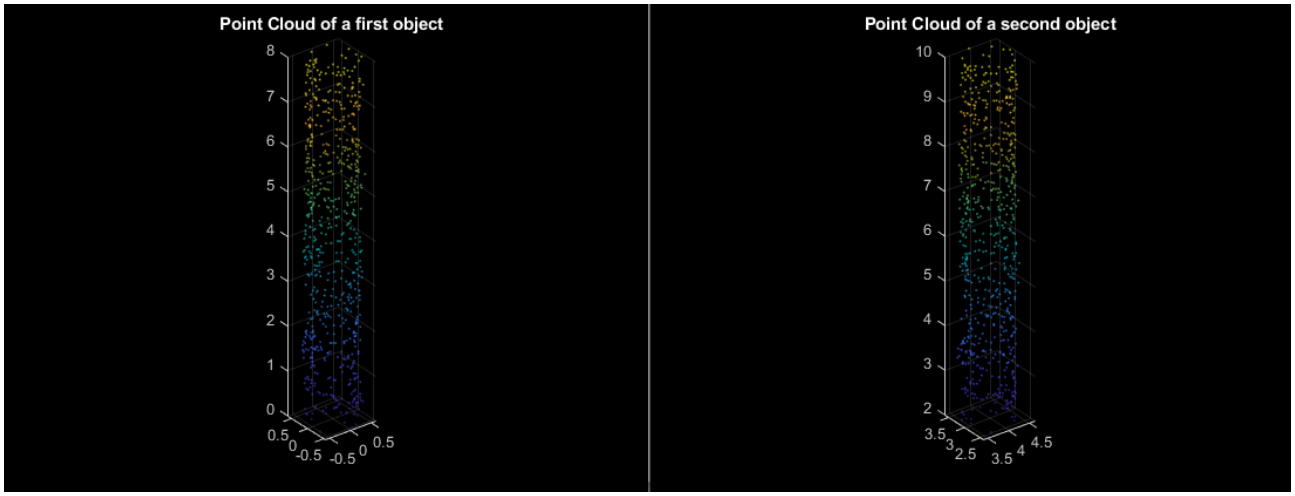

Рис. 4.3 – Сгенеровані шаблони шуканих об’єктів у тривимірному просторі

```
figure
pcshowpair(ptCloud1,ptCloud2)
title('Point Clouds After Rotation')
```

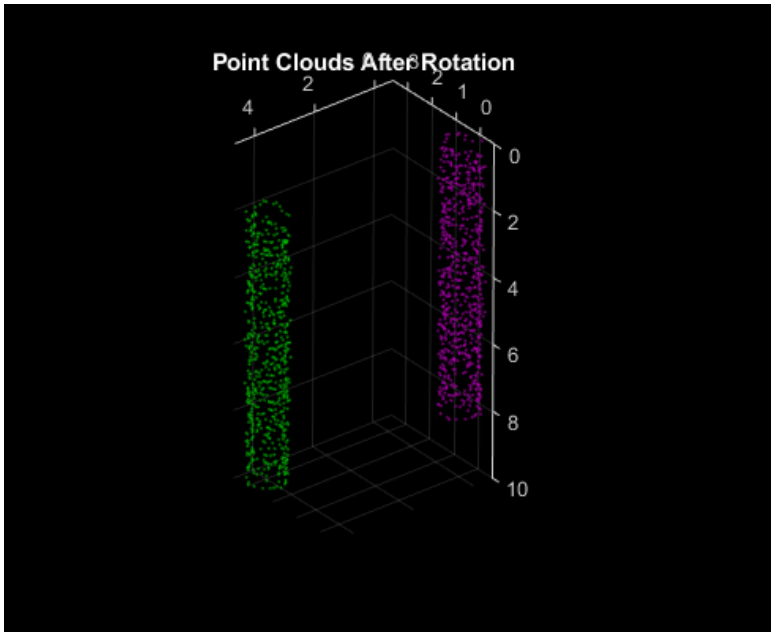

Рис. 4.4 - Зіставлення двох об’єктів у просторі

```
%{ Added Noise}
noise1 = rescale(rand(300,3),-2,2);
ptCloudWithNoise1 = pointCloud([ptCloud1.Location;noise1]);
figure
pcshow(ptCloudWithNoise1);
title('Point Cloud 1 with Noise')
noise2 = rescale(rand(300,3),-2,2);
ptCloudWithNoise2 = pointCloud([pctransform(ptCloudWithNoise1,tform).Location]);
figure
pcshow(ptCloudWithNoise2);
title('Point Cloud 2 with Noise')
```

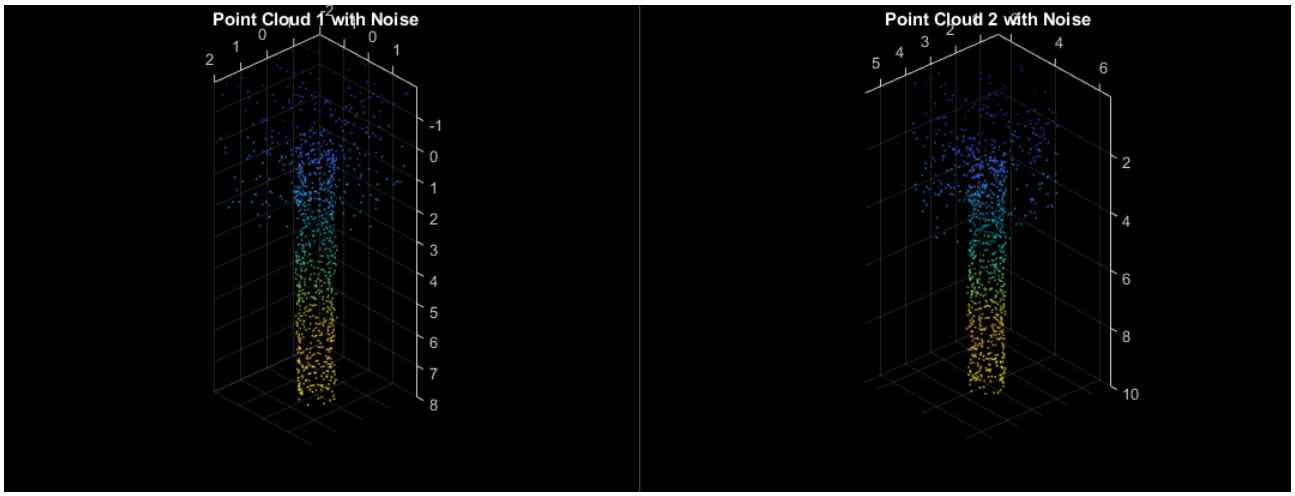

Рис. 4.5 Створення додаткових шумів для об’єктів, котрі відповідають погрішностям при пошуку об’єкту на сцені

```
figure
pcshowpair(ptCloudWithNoise1,ptCloudWithNoise2)
title('Point Clouds After Rotation and Move With Added Noise')
```

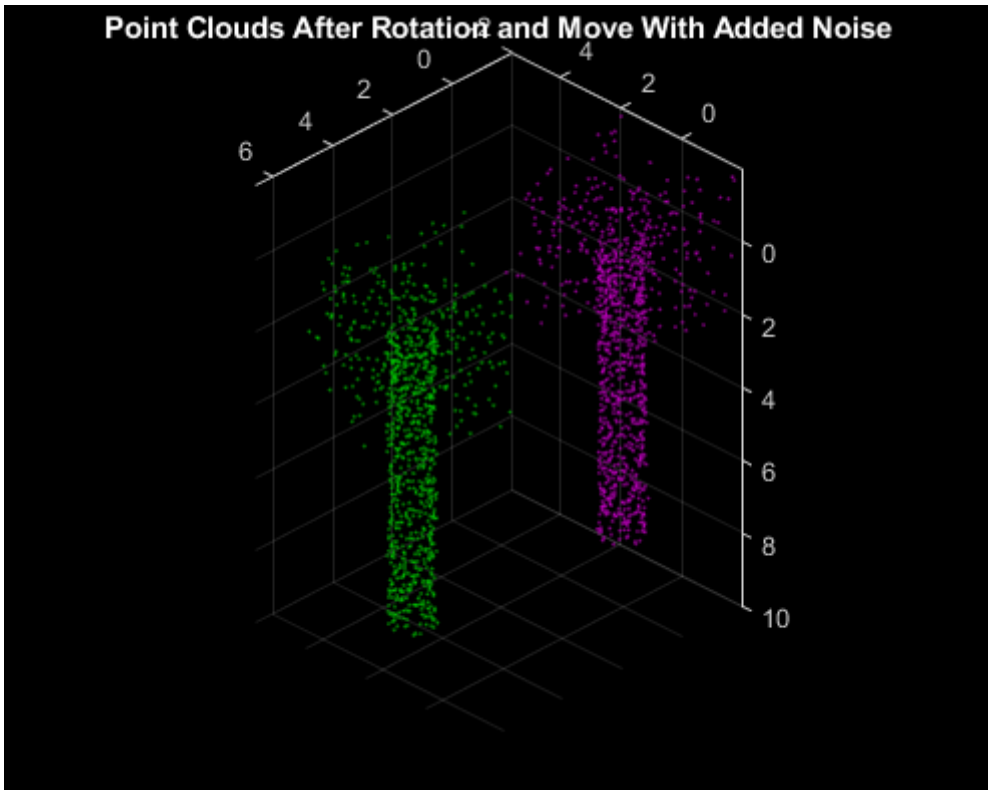

Рис. 4.6 - Зіставлення двох об’єктів у просторі після додавання шумів

```
% { Extract matched points from the point clouds }
```

```
matchedPoints1 = ptCloud1.Location;
matchedPoints2 = ptCloud2.Location;

% { Estimate the similarity transformation (affine) between two point clouds }
[tformEst,inlierIndex] = estimateGeometricTransform3D(matchedPoints1,
matchedPoints2, 'similarity');
inliersPtCloud1 =
transformPointsForward(tformEst,matchedPoints1(inlierIndex,:));
inliersPtCloud2 = matchedPoints2(inlierIndex,:);

maxDistance = 0.2;
model1 = pcfitcylinder(ptCloud1, maxDistance);
model2 = pcfitcylinder(ptCloud2, maxDistance);

figure;
subplot(1,2,1);
plot(model1)
hold on;
subplot(1,2,2);
plot(model2)
hold on;
suptitle('model');

[optimizer, metric] = imregconfig('multimodal');
optimizer.InitialRadius = 0.009;
optimizer.Epsilon = 1.5e-4;
optimizer.GrowthFactor = 1.01;
optimizer.MaximumIterations = 300;
rezMatrix = tfoemEst.T;
rezMatrix = tformEst.T;
disp('Affine Transformation matrix: '); fprintf( '%d;%d;%d;%d;%d\n',
rezMatrix');
```

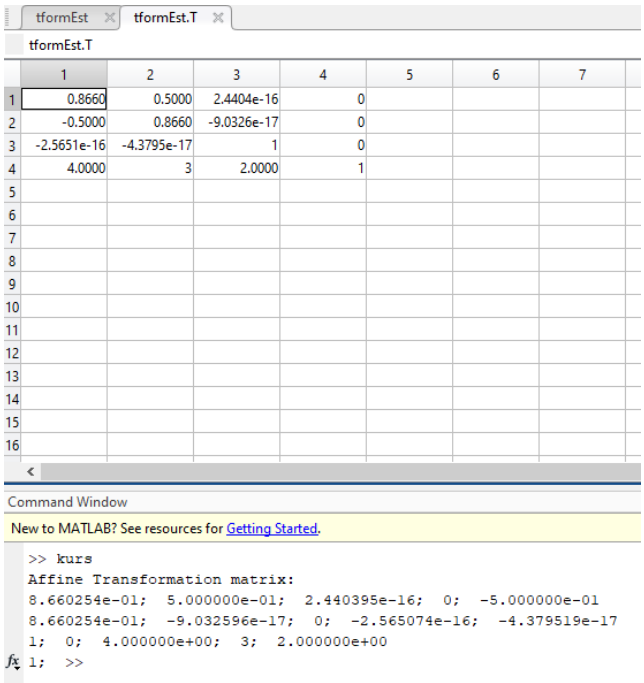

Рис. 4.7 – Шукана матриця перетворення

ВИСНОВКИ

Під час виконання магістерської роботи було розглянуто принципи детектування та відображення руху об’єктів у просторі за допомогою Афінних перетворень. Після аналізу принципів перетворень, як тривимірних, так і двовимірних, було розглянуто матриці основних перетворень - обертань, розтягнень, віддзеркалень і переносів. Проаналізувавши комбінацій цих перетворень і наклавши їх на тривимірну модель було визначено матрицю афінних перетворень. Примінивши цю матрицю до первісних координат апарату визначається, як саме змінилося положення БПЛА у просторі.

Якщо намагатися робити всі обчислення апаратним ПЗ значно підвищіться його автономність, проте в той же час це призведе до збільшення його обчислювальних можливостей\батареї, а отже й до збільшення габаритів самого апарату. Альтернативою була б передача лише матриці перетворення на обладнання оператора, а це в свою чергу потребуватиме його постійного перебування недалеко від БПЛА. Оптимальну середину треба необхідно знаходити залежно від поставленої перед апаратом задачі.

Також значну перевагу можна отримати, збільшивши точність тривимірної моделі території, на якій відбуватиметься політ – це допоможе оператору краще орієнтуватися на місцевості (якщо данні надсилаються напряму йому), або ж додати додаткові шукані об’єкти, що дозволить дозволить аналізувати одразу декілька матриць перетворення та робити висновки що до положення апарату на основні кількох джерел (якщо від апарату потребується максимальна самостійність).

СПИСОК ЛІТЕРАТУРИ

1. В.Н. Козлов. Элементы математической теории зрительного восприятия – 2001. – P. 116–120.
2. Скуднев О.В., Дубограй И.В. Применение аффинных преобразований в моделировании механических систем на примере расчёта навигационных параметров гироскопических приборов – 2017. – P. 92–100.
3. А.В. Ершов. Линейные и Аффинные пространства и отображения – 2016. – P. 38–48.
4. Mikolajczyk K., Schmid C. A performance evaluation of local descriptors // Proc. IEEE Conference on Computer Vision and Pattern Recognition. – 2003. – P. 257–264.
5. Hu M. K. Visual pattern recognition by moment invariants // IRE Trans. on Information Theory. – 1962. – V.8. – P. 179–187.
6. Flusser J. On the independence of rotation moment invariants // Pattern Recognition. – 2000. – V. 33. – P. 1405–1410.
7. Bay H., Ess A., Tuytelaars T., Van Gool L. SURF: Speeded up robust features // Computer Vision and Image Understanding. – 2008. – V. 110. – P. 346–359
8. Tarrida, Agusti R., Affine spaces, Affine Maps, Euclidean Motions and Quadrics. – 2011. – P. 16–21.
9. Dolgachev, I.V.; Shirokov, A.P., Affine space – 2001. – P. 93–97.
10. Snapper, Ernst; Troyer, Robert J., Metric Affine Geometry – 1989. – P. 13–21.
11. Okhrimenko K.P., Information system for determining the coordinates of an aircraft in the absence of a global positioning service (GPS) – 2019. – P. 13-16

Додатки

MATLAB

```
%{ read object and scene }%
I1 = imread('pik1.jpg');
objImage1 = rgb2gray(I1);
dicomwrite(objImage1, 'pik1.dcm');
fixed = dicomread('pik1.dcm');

I2 = imread('pik2.jpg');
objImage2 = rgb2gray(I2);
dicomwrite(objImage2, 'pik2.dcm');
moving = dicomread('pik2.dcm');

figure;
subplot(1,2,1);
imshow(fixed);
hold on;
subplot(1,2,2);
imshow(moving);
hold on;
imshowpair(fixed, moving,'Scaling','joint')
suptitle('.jpg to .dcm');

%{ generate first object and move it for some distance and rotating the data
points around x axis}%
[theta, r, h] = meshgrid(0:.1:6.28, 0.5, 0:.2:8);
r = r + 0.05 * randn(size(r));
[x, y, z] = pol2cart(theta, r, h);
P1 = ([x(:), y(:), z(:)]')';
P2 = (rotx(60) * [x(:), y(:), z(:)]')';

scatter3(P1(:, 1), P1(:, 2), P1(:, 3))
scatter3(P2(:, 1), P2(:, 2), P2(:, 3))
axis equal

pC1 = pointCloud(P1);
pC2 = pointCloud(P2);

%{ create some generic transformation }%
theta = 30;
rot = [cosd(theta)  sind(theta) 0; ...
      -sind(theta)  cosd(theta) 0; ...
            0           0       1];
trans = [4 3 2];
tform = rigid3d(rot,trans);

figure
pcwrite(pC1,'ptCloud1.ply','Encoding','ascii');
ptCloud1 = pcread('ptCloud1.ply');
ptCloud1 = pcdownsample(ptCloud1,'random',0.25);
pcshow(ptCloud1);
title('Point Cloud of a first object')

figure
ptCloud2 = pctransform(ptCloud1,tform);
pcshow(ptCloud2);
title('Point Cloud of a second object')
```

```
figure
pcshowpair(ptCloud1,ptCloud2)
title('Point Clouds After Rotation')

 %{ Added Noise}
noise1 = rescale(rand(300,3),-2,2);
ptCloudWithNoise1 = pointCloud([ptCloud1.Location;noise1]);
figure
pcshow(ptCloudWithNoise1);
title('Point Cloud 1 with Noise')
noise2 = rescale(rand(300,3),-2,2);
ptCloudWithNoise2 = pointCloud([pctransform(ptCloudWithNoise1,tform).Location]);
figure
pcshow(ptCloudWithNoise2);
title('Point Cloud 2 with Noise')

figure
pcshowpair(ptCloudWithNoise1,ptCloudWithNoise2)
title('Point Clouds After Rotation and Move With Added Noise')

% { Extract matched points from the point clouds }
matchedPoints1 = ptCloud1.Location;
matchedPoints2 = ptCloud2.Location;

% { Estimate the similarity transformation (affine) between two point clouds }
[tformEst,inlierIndex] = estimateGeometricTransform3D(matchedPoints1,
matchedPoints2, 'similarity');
inliersPtCloud1 =
transformPointsForward(tformEst,matchedPoints1(inlierIndex,:));
inliersPtCloud2 = matchedPoints2(inlierIndex,:);

maxDistance = 0.2;
model1 = pcfitcylinder(ptCloud1, maxDistance);
model2 = pcfitcylinder(ptCloud2, maxDistance);

figure;
subplot(1,2,1);
plot(model1)
hold on;
subplot(1,2,2);
plot(model2)
hold on;
suptitle('model');

[optimizer, metric] = imregconfig('multimodal');
optimizer.InitialRadius = 0.009;
optimizer.Epsilon = 1.5e-4;
optimizer.GrowthFactor = 1.01;
optimizer.MaximumIterations = 300;

rezMatrix = tformEst.T;
disp('Affine Transformation matrix: '); fprintf( '%d;  %d;  %d;  %d;  %d\n',
rezMatrix');
```