ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра комп'ютерних наук

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «Визначення координат фотокадру знятого з БПЛА в режимі реального часу»

Виконав: студент 4 курсу, групи КНД–41

спеціальності 122 Комп'ютерні науки

Устименко Н. С.
(прізвище та ініціали)

Керівник Вишнівський В.В.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Нормоконтроль
(прізвище та ініціали)

Київ – 2021

ВСТУП

Актуальність роботи полягає в економічній вигоді та використанні менших ресурсів для проведення аерофтозйомки з використанням БПЛА та відкитті нових шляхів обробки даних повітряної зйомки.

Метою дослідження є оптимізація та підвищення ефективності процесів керування, аерофотозйомки та обробки навігаційних даних з подальшим їх використанням в областях картографії, моніторингу навколишньої серди, тощо.

Об’єктом дослідження є процес отримання точних координат фотокадрів та супутніх процесі, оптимізація та автоматизація задач управління та навігації безпілотних літаків.

Предмет дослідження – обробка навігаційних даних та даних льотних завдань.

Наукова новизна дослідження полягає в малій кількості літератури присвяченої саме темі розрахунку координат аерофотозйомки з малих повітряних судів, так як такий вид зйомки важко назвати задовілним для потреб картографії.

1. АНАЛІЗ СИСТЕМ ФОТОЗЙОМКИ З БЕЗПІЛОТНОГО ЛІТАЛЬНОГО АПАРАТУ. ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1. Огляд процесу фотозйомки з безпілотного літального апарату

Застосування безпілотних літальних апаратів (БПЛА) дозволяє істотно знизити витрати на виробництво аерофотознімальних робіт. З точки зору традиційної фотограмметрії якість подібної зйомки найімовірніше буде оцінено, як неприйнятне, оскільки на БПЛА, як правило, встановлюються камери побутового сегмента, не використовується гіростабілізуюча апаратура, при зйомці нерідкі відхилення оптичних осей від вертикалі в кілька градусів, що значно ускладнює процес первинної обробки знімків.

З цього виходить одна з основних проблем визначення координат фотокадру аерофотозйомки – спотворення центральної проекції кадру. Через те, що площина матриці фотокамери майже ніколи не паралельна до площини поверхні землі в наслідок технічних та природних обурень (зміна курсу, вітер і т.д.).

Для подальшого розгляду процесів аерофотозйомки та впливів різних обурень на якість та придатність фотокадрів до визначення точних координат та використання їх для формування топографічних карт, слід визначити, що таке фотознімок, центральна проекція та їх властивості.

Фотознімок – це зображення, яке відповідно до законів геометрії, є перспективним, побудованим в центральній проекції, в якій все проміння світла, відображені від об’єкту фотозйомки, проходять через точку, звану центром проекції[15].

Основними елементами центральної проекції є:

1. Картинна площина або площина знімка
2. Центр проекції

3. Площина фотографованої області

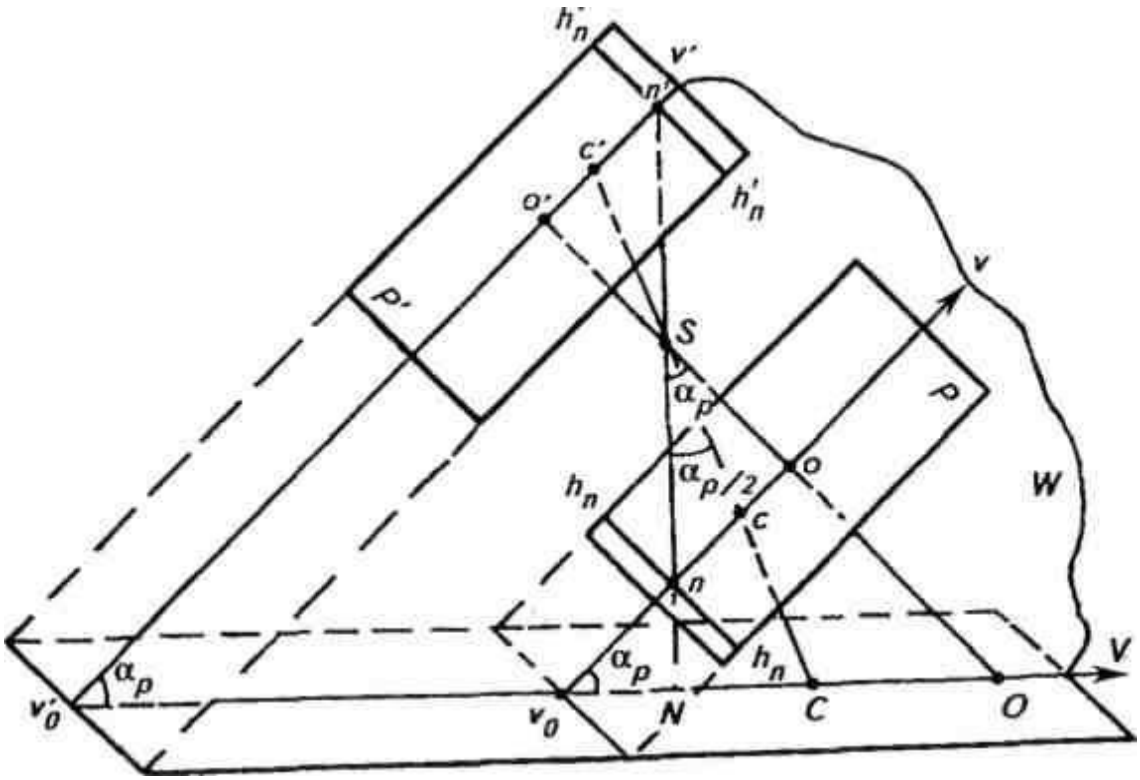


Рис. 1.1 – Основні елементи центральної проекції

Основні елементи центральної проекції:

E - предметна площина - горизонтальна січна плос-кістка ділянки місцевості, що знімається;

o(o') - головна точка картини - головна точка знімка, получа-емая при перетині головного променя(оптичній осі) об'єктиву знімальної камери S0 з площиною картини;

W - площина головного вертикала, що проходить через точку S перпендикулярно площинам P(P') і E;

VoV(VoV') - головна вертикаль - слід перетину площин P(P') і W; v0V - проекція головної вертикалі;

n(n') - точка надіра - точка перетину площини P{P') з від-весным променем; N - проекція точки надіра -точка перетину площини E прямовисним променем, що проходить через точку S;

ap - кут нахилу картини(знімка) - кут між плоскостя-мі P(P) і E або променями SO і SN;

c(c) - точка нульових спотворень - точка перетину плоско-сти P(P) бісектрисою кута aP;

C - проекція точки нульових спотворень; hnhn(h'nh'n) - горизонталь, що проходить через точку n(n'), -ли-ния в площині P(P'), перпендикулярна v0v(VoV').

Проектуючий промінь, перпендикулярний до площини знімка та збігається з оптичною віссю об’єктива фотоапарату, називається головним променем. Також, всі промені, які входять в центр проекції від точок об’єкту та пертинають площину знімка називаються проектуючими променями[15].

Але, слід відзначити, що для сучасного фотограмметричного програмного забезпечення ці недоліки не представляють значних проблем. Більш того, розвиток цифрових методів фотограмметричної обробки вже призвело до появи програм і програмних комплексів, здатних обробляти навіть такі "неякісні" дані аерофотозйомки в високоавтоматизованому режимі, при мінімальній участі оператора.

Розглянемо технологічний ланцюжок отримання топографічної карти з використанням наступних компонентів:

• БПЛА для виконання аерофотозйомки;

• ПО Agisoft PhotoScan як інструмент обробки матеріалів зйомки;

• інструментарій ГІС Панорама для векторизації ортофотопланів та отримання топокарт.

Аерофотозйомка з використанням БПЛА

З технічної точки зору процес аерофотозйомки з використанням БПЛА складається з трьох етапів: підготовчого, власне зйомки, і постобробки отриманих даних.

Підготовчий етап

На даному етапі проводиться:

• вивчення наявних матеріалів; формування або збір вимог до матеріалів, які потрібно отримати за результатами зйомки - тип і масштаб

карти, межі об'єкта зйомки; приведення їх у технічні вимоги до знімальним матеріалами: дозвіл, координати контуру ділянки зйомки, перекриття знімків, точність визначення координат центрів фотографування, вимоги до наземної опорної мережі (при комбінованому зніманні, наприклад, коли прив'язка фотоплана проводиться по точках наземної опорної мережі, вимоги до точності визначення КЦФ взагалі не пред'являються);

• формування польотного завдання для БПЛА. Виконується програмою - планувальником польоту, що входить до складу комплексу. Оператор повинен вибрати використовуваний комплекс БПЛА (в разі, якщо програма дозволяє працювати з декількома конфігураціями БПЛА і фотоапаратури), задати на карті контур ділянки зйомки і зразкове положення стартового майданчика, встановити необхідний дозвіл і перекриття, після чого програма розраховує план польоту і перевіряє його здійсненність .

Виконання аерофотозйомки

Після прибуття на стартовий майданчик виробляється:

• уточнення положення стартового майданчика, завдання точки повернення і введення даних про швидкість і напрям вітру на робочій висоті, якщо такі відомі;

• автоматичне уточнення плану польоту і повторна перевірка його здійсненності;

• старт БПЛА з пускового пристрою;

• виконання зйомки в автоматичному режимі;

• посадка.

Виконання зйомки місцевості з використанням БПЛА

При використанні комбінованого способу виконується визначення координат опорних точок, вибраних для прив'язки.

Постобробка даних

Полягає в:

• зняття даних (фотознімки та журнал польоту) з бортових носіїв інформації;

• візуальною оцінкою якості фотографій и відбракування "технічних" кадрів. Під технічними кадрами розуміються знімки, зроблені поза межами ділянки зйомки - при підльоті до ділянки, на дугах розворот і т.п.;

• генерація файлу прив'язки центрів фотографування. В ході польоту апаратура управління веде запис різних параметрів, серед яких - координати, швидкість і параметри орієнтування літального апарату. Після закінчення зйомки з файлу журналу польоту необхідно вибрати координати, що відповідають моментам фотографування, і приписати їх конкретним знімкам. Така обробка, як правило, виконується в тій же програмі - за розкладом польотного завдання.

Відповідно до вимог галузевих інструкцій [1], для отримання топокарт масштабу 1: 2000 необхідна фотооснова, що має дозвіл 15 см / пікселів і має похибку визначення координат в кожній точці не вище 60 см. Такий дозвіл легко забезпечується при зйомці з БПЛА з використанням компактних фотоапаратів. Наприклад, зйомка камерами типу Canon S-95 або Sony NEX-5 (з об'єктивом SEL30M35) з висоти близько 200-300 м дає знімки, які мають дозвіл 5 см / піксель.

Прив'язка необхідної точності досягається вимірюванням координат центрів фотографування з використанням високоточних GNSS-приймачів в межах референцній мережі, або залученням наземної опорної мережі, точки якої прив'язані з похибкою не вище 30 см.

1.2. Огляд наявних систем фотозйомки з безпілотного літального апарату

Обробка аерофотознімків в ПО Agisoft PhotoScan

Програма Agisoft PhotoScan - універсальний інструмент для генерації тривимірних моделей поверхонь об'єктів зйомки по фотозображенню цих об'єктів. PhotoScan з успіхом застосовується як для побудови моделей

предметів и об'єктів різних масштабів - від мініатюрних археологічних артефактів до великих будівель і споруд, так і для побудови моделей місцевості за даними аерофотозйомки і генерації матриць висот и ортофотопланів, побудованих на основі цих моделей. Обробка даних в PhotoScan гранично автоматизована - на оператора покладаються лише функції контролю і управління режимами роботи програми.

Побудова и прив'язки моделі місцевості в Програмі складається з трьох основних етапів:

• побудова грубої моделі. На цьому етапі проводиться автоматичне визначення спільних точок на перекриваючих знімках, відновлення проектуючих променів, визначення координат центрів фотографування і елементів взаємного орієнтування знімків, розрахунок параметрів, що описують оптичну систему (дисторсія, коефіцієнт асиметрії, положення центральної точки). Всі ці розрахунки виконуються в програмі за одну операцію;

• прив'язка отриманої моделі до зовнішньої (геодезичної, географічної) системі координат і зрівняння всіх параметрів системи - координат центрів фотографування і наземних опорних точок, кутів орієнтування знімків, параметрів оптичної системи з використанням параметричного методу зрівнювання. Як вагових коефіцієнтів для зрівнювання виступають похибки визначення координат точок зйомки (центрів фотографування), визначення координат точок наземної опорної мережі, дешифрування і маркування опорних точок на знімках;

• побудова полігональної моделі поверхні місцевості на основі певних, на попередньому етапі, параметрів. У програмі реалізований експрес-спосіб, що полягає в тріангуляції тільки спільних точок, отриманих на першому етапі, і більш точні способи обробки, які полягають у визначенні просторового положення для кожного пікселя зображення (в залежності від заданого ступеня деталізації обробляється

кожен перший, кожен четвертий, кожен шістнадцятий , і т. д. - всього п'ять можливих рівнів).

Потім отримана модель використовується для генерації ортофотопланів та матриць висот.

З точки зору оператора процес роботи з програмою виглядає наступним чином:

1. Завантаження фотознімків

2. Вибір системи координат і завантаження даних прив'язки центрів фотографування

3. Формування точкової моделі поверхні Землі

4. При наявності наземної опорної мережі - установка відміток опорних точок на фотознімках і завантаження координат точок опорної мережі

5. Оптимізація моделі (зрівнювання параметрів прив'язки)

6. Генерація полігональної моделі поверхні Землі

7. Експорт даних - ортофотоплан, матриця висот.

Безпосередньо в графічному інтерфейсі програми можна виробляти базові вимірювання на отриманій моделі - вимірювати відстані, площу поверхні і об'єм моделі.

Розвинений API дозволяє створювати скрипти на мові Python, керуючі обробкою і відображенням даних, що дозволяє ще більше автоматизувати рішення типових задач.

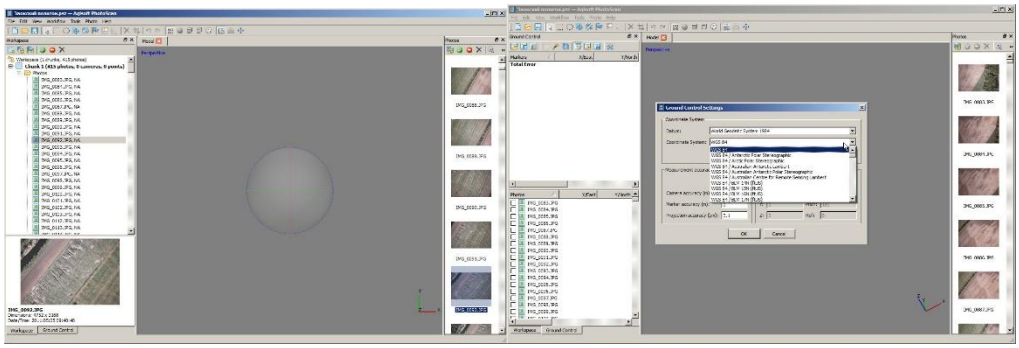


Рис. 1.2 - Фотографії завантажені

1)У властивостях проекту видно, що проект складається з блоків (chunks) - оброблюваних незалежно частин проекту зі своїми фотографіями, моделлю, СК, параметрами калібрування оптики і т.п. В даному проекті - один блок, що складається з 415 фотографій. Мітки NA (not aligned) поруч з фотографіями показують, що положення цих знімків в просторі моделі ще не відомо.

2) Вибір системи координат

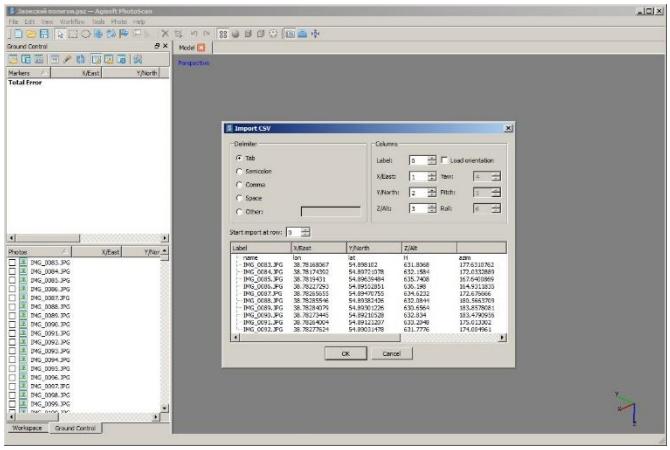


Рис. 1.3 - Вибір системи координат

Завантаження даних телеметрії - координат центрів фотографування (КЦФ). Програма розпізнає дані в форматах txt, csv, tel і дозволяє вказати з яких стовпців зчитувати дані

4) Мітки в формі синіх кульок відображають взаємне розташування точок зйомки (КЦФ), після зрівнювання вони будуть замінені знаками іншого виду, відповідним положенням площин кадрів

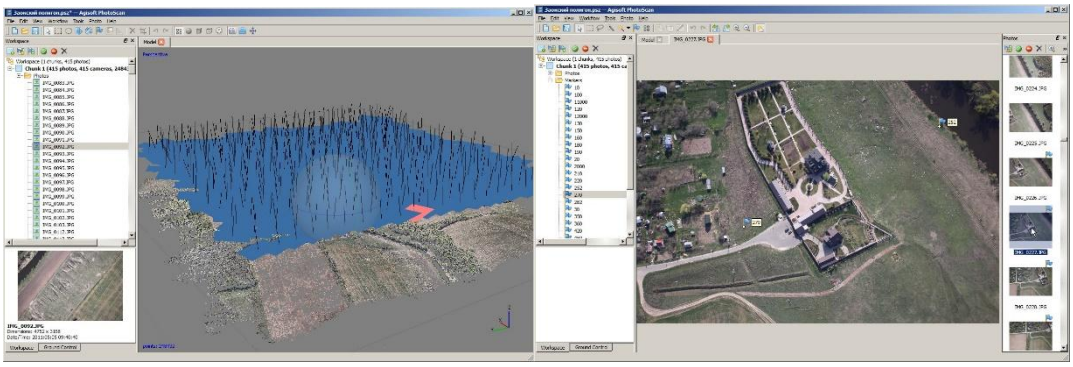


Рис. 1.4 - Вибір міток

5) Після виконання першого етапу обробки - первинного зрівнювання і побудови точкової моделі, формується хмара точок, що описує модель, і набір параметрів взаємного орієнтування знімків. Положення Вибране зображення відображається в області перегляду моделі. Знімки, які не вдалося зрівняти, як і раніше відображаються сферами / кульками і в списку фотографій відзначені міткою NA (not aligned). В даному проекті таких немає.

6) Установка маркерів (міток опорних точок). Якщо відомо положення маркерів на знімках (в системі координат знімка), можна просто імпортувати ці дані в PhotoScan. Якщо маркери ще не розшифровані, доведеться ставити їх розташування прямо в програмі. Для кожного маркера досить відзначити їх положення на одному-двох знімках, і PhotoScan автоматично визначає їхнє положення на інших знімках, виділяючи знімки, на яких присутній вибраний маркер, спеціальними позначками. На кожному знімку можна підтвердити або уточнити автоматично вибране положення маркера

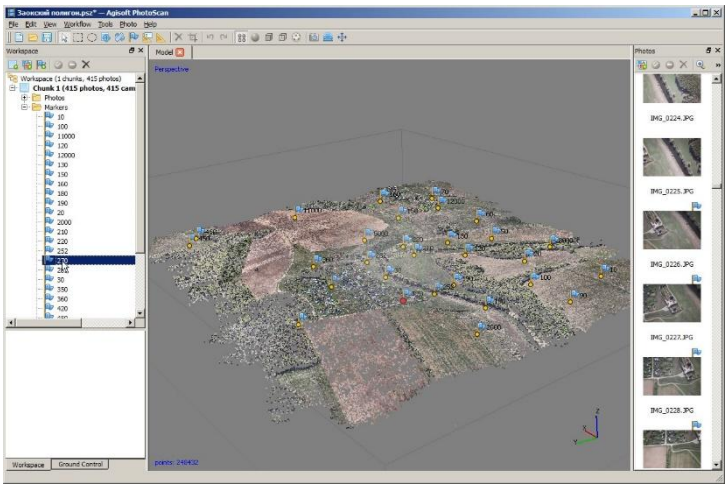


Рис. 1.5 - Вибір маркерів

7) Маркери розставлені. Можна виконувати побудову моделі місцевості

8) Модель готова. Її можна експортувати як матрицю висот (цифрову модель місцевості) і сформувати на основі цієї моделі ортофотоплан місцевості.

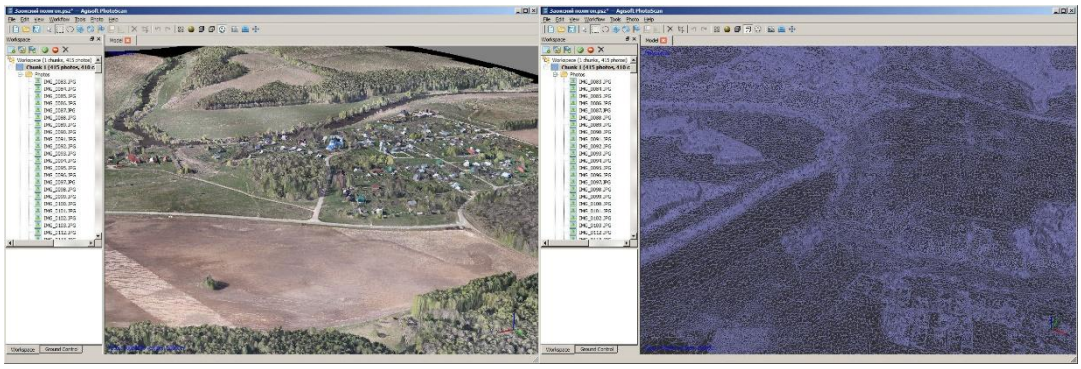


Рис. 1.6 - Готова модель

9) На завершення можна побудувати текстуру моделі і розглядати її прямо в програмі.

10) Внутрішнє представлення моделі поверхні Землкі в PhotoScan - мережа тріангуляції Делоне, TIN модель

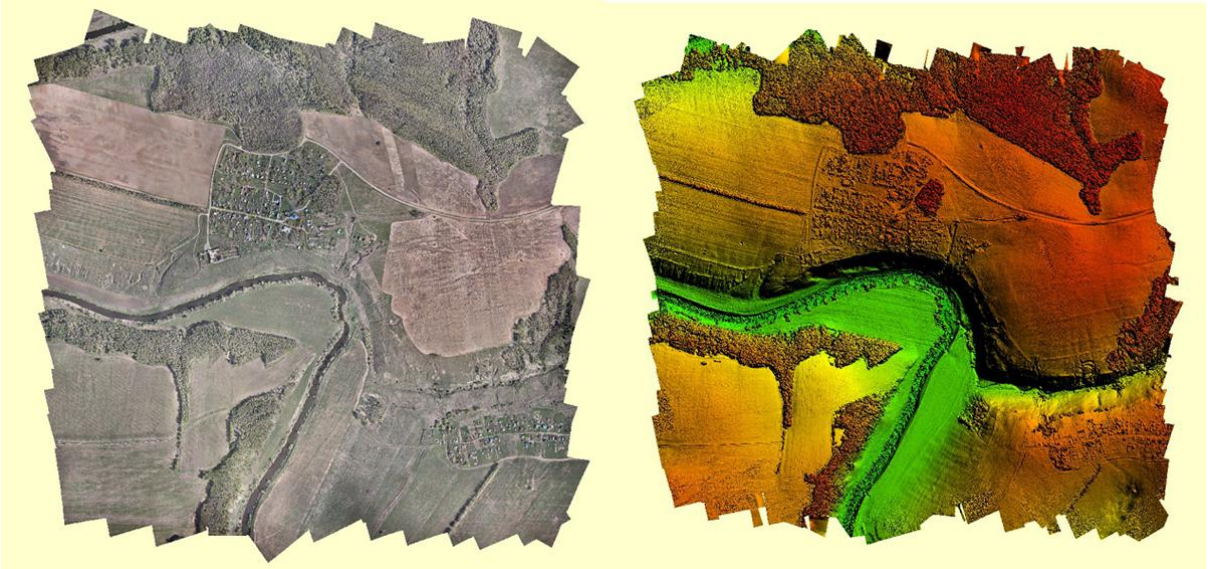


Рис. 1.7 - Мережа триангуляції Делоне, TIN модель

11) Ортофотоплан всієї ділянки зйомки.

12) Цифрова модель місцевості всієї ділянки зйомки.

Також розглянемо метод вирішення поставленого завдання вручну. Так, як ми з'ясували, що знімок - це зображення, що є перспективним, розглянемо перспективу на прикладі точки *A*, що належить площині *T*. Для цієї точки необхідно знайти перспективне розташування в площинні фотокадру *P*. Відтворимо зображення відрізків ліній на фотокадрі *P*, які знаходяться в площині основи, які паралельні напрямку проведення знімання. На рис. 2.3 зображені відрізки AA_1 та BB_1 , що є паралельними напрямку VV_1 [15].

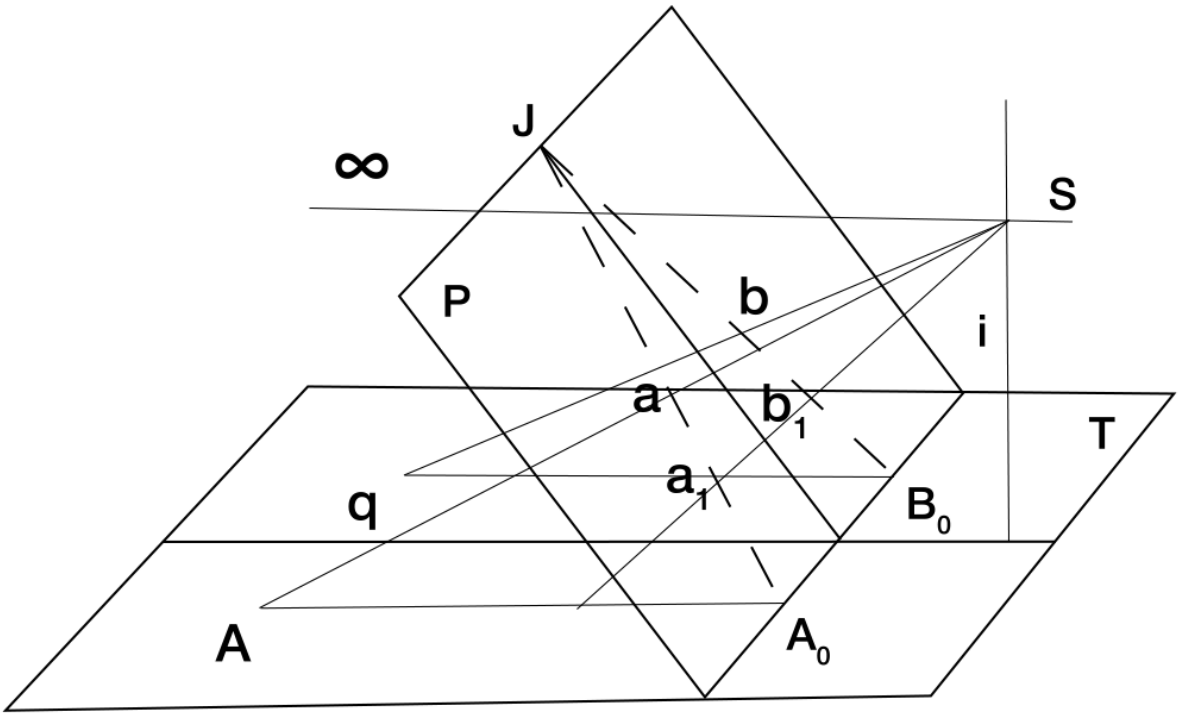


Рис. 1.8 – Зображення перспектив точок і відрізків на площині фотокадру *P*

Проводячи через центр заданої проекції *S* горизонтальний промінь в площині *Q*, отримуємо точку сходження *J* в площині фотокадру *P*.

Проводячи перпендикуляри з точок A і B на перспективну вісь $tt`$, то отримуємо A_0, B_0 – що є точками перетину. Ці точки є з'єднаними з точкою сходження J. З даних точок A, A_1, B, B_1 промені направлені в точку S. Перетин цих проектуючих відрізків з лініями JA_0, JB_0 , показує перспективне зображення так точок, як a, a_1, b, b_1 и ліній aa_1 та bb_1.

Розглянемо також перспективу горизонтального відрізка прямої, з якої складається кут за напрямком фотографування $VV`$. В площині $T$ знаходиться горизонтальний відрізок $AA_1$ (рис 2.4). Потрібно визначити його перспективне зображення в площині фотокадру $P$. Продовженням відрізку $AA_1$ до його перетину з віссю $tt`$, на якій одержано точку A_0. З центральної точки проекції S проводимо промінь паралельний відрізку AA_1 і на осі дійсного горизонту отримаємо отримуємо точу i. Промінь iA_0 є напрямом перспективи частини прямої AA_1 в площині P. Перетин променя iA_0 з променями SA і SA_1 надає перспективне зображення aa_1 від даного відрізка[15]. Віддаленість точки i від головної точки перентину J можна обчислити за формулою

$$Ii = f\frac{tg\varphi}{\sin\alpha_0} \qquad (1.1)$$

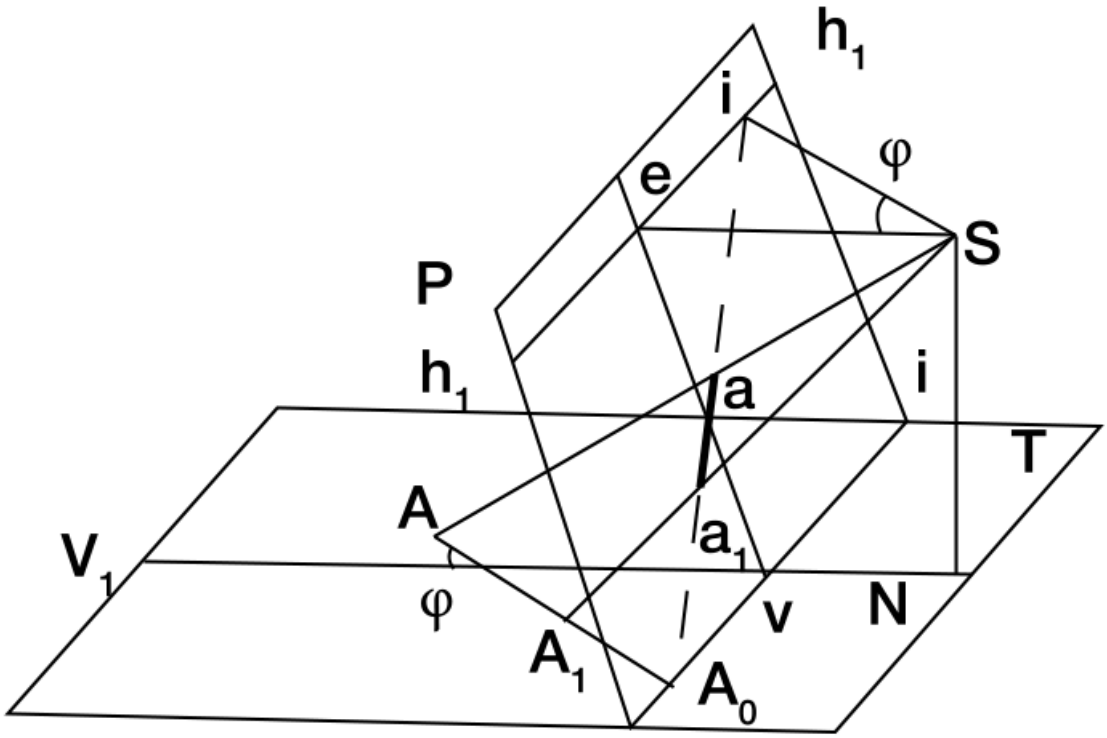


Рис. 1.9 – Відтворення на площині *P* прямої, яка складає кут з площиною земної поверхні

На рис. 2.5 зображено перспективне відображення відрізка на площині *T*, яке є перпендикулярним до прямої *VV'*. Відрізок aa_1 на площі кадру *P* перпендикулярний до головної вертикальної прямої фотокадру.

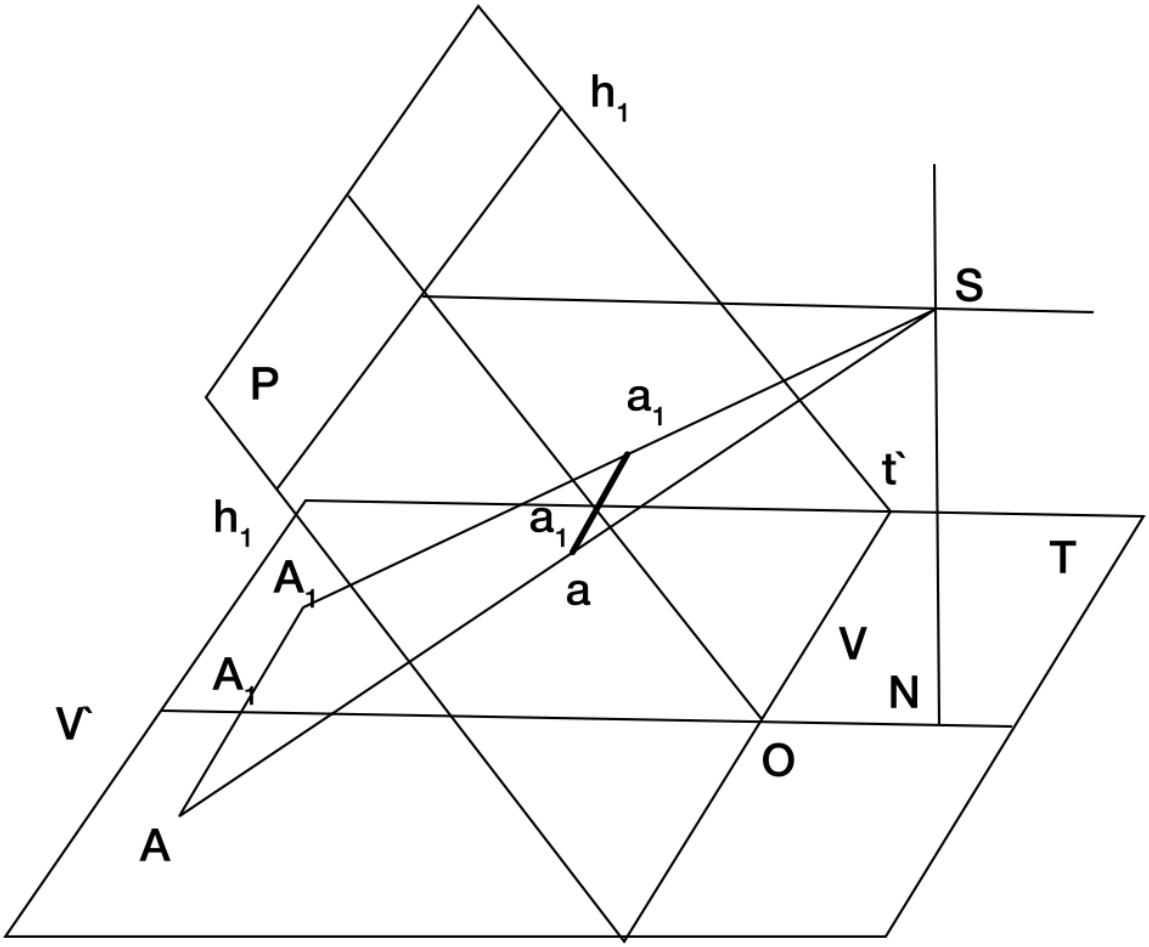


Рис. 1.10 – Зображення відрізка прямої, що перпендикулярна до напряму зйомки

Прямовисна лінія *AB* є перпендикулярною площині *T*. Для віднайдення її перспективного відображення в площі фотознімка БПЛА проведемо пряму *SN*, яка буде паралельна лінії *AB* і через обидві лінії побудуємо площу *Q* (Рис. 2.6). Вертикальна площина *Q* буде перетинати предметну площу *T* за лініями *BN* і на перспективній осі *tt*’ здобудемо точку *I*.

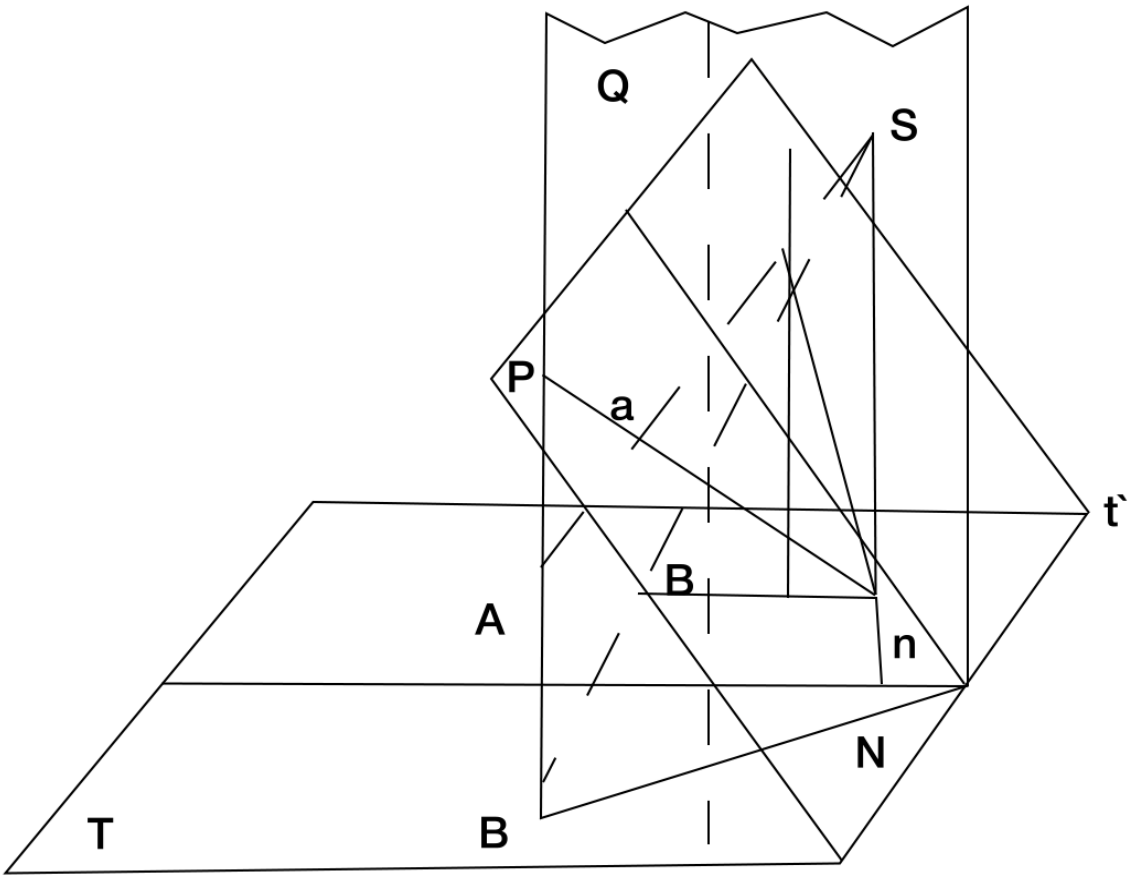


Рис. 1.11 – Перспективне відтворення прямовисних відрізків

Точкою перетину зображення на площині фотокадру *P* вертикальних відрізків є точка надір *n*. Звертаючи увагу на те, що точки *l* та *n* розташовані в площі *P*, то шуканий відрізок *ав* знаходиться на відрізку *ln*. Схід променів *SA* і *SB* з променем *ln* надає відображення відрізків *ав*. Довжина частини прямої *ав* складає перевищення між шуканими точками земної площини.

Ця методика використовується для вирішення поставленого завдання вручну. На основі даної методики буде розроблена математична модель для визначення координат точок фотогрфії.

1.3. Постановка задачі дослідження

Суть дослідження полягає в розробці системного комплексу автоматизованої обробки навігаційних даних, знімків БПЛА, прив’язки координат центрів фотографування та визначаення кооординат крайніх точок фотокадру з урахування спотворення центральної проекції у наслідок природних та технічних обурень за даними ДЗЗ.

Питання обробки і аналізу зображень при вирішенні різних завдань спостереження з БПЛА широко висвітлені у вітчизняній і зарубіжній літературі. Однак проблеми, пов'язані з отриманням координат фотознімку спостереженнями з безпілотних літальних апаратів, що необхідно для оптимального планування маршрутів руху, режимів роботи та навігації техніки, приділяється недостатньо уваги [1]. Метою навігації БЗТ(безпілотної збиральної техніки) є находження оптимальних (відповідно до заданих критеріїв) маршрутів її переміщення між заданими точками простору з урахуванням перешкод та картограм [2]. Виходячи із цього, розробка нових моделей, методів і алгоритмів отримання координат фотозйомки за допомогою безпілотних літальних апаратів є актуальною науково-технічною проблемою.

Процес планування змісту та часу планування траєкторій руху поділяється на декілька етапів, а саме: формування електронної карти місцевості, картограм з координатами, визначення всіх видів перешкод на кожній ділянці з БПЛА та розрахунок оптимальних маршрутів руху і режимів роботи на основі застосування методів статистичного аналізу.

Значну роль у вирішенні цієї задачі відіграють спеціалізовані геоінформаційні системи (ГІС), що враховують просторову прив'язку. Як показує практика, зйомки з БПЛА не тільки підвищують точність, однорідність, об'єктивність і частоту спостережень.

Для ефективного керування необхідне рішення багатьох складних завдань, серед яких збір даних з БПЛА та їх обробка за допомогою спеціального програмного забезпечення.

Для вирішення задач аналізу, прогнозу і оптимізації керування геоінформаційні системи повинні включати цифрові карти схилу з моделлю рельєфу місцевості та експозицій схилів, розташування ліній електропередач та інших стаціонарних перешкод на шляху руху.

Таким чином, сформовані рекомендації використовуються для планування збиральних робіт та визначення оптимальних маршрутів руху і режимів роботи БЗТ з урахуванням багатьох критеріїв оптимальності, таких як зменшення витрат пального, небезпечного об’їзду перешкод.

2. ПРОЕКТУВАННЯ СИСТЕМИ ВИЗНАЧЕННЯ КООРДИНАТ ФОТОКАДРУ ЗНЯТОГО З БЕЗПІЛОТНОГО ЛІТАЛЬНОГО АПАРАТУ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

2.1. Методика визначення координат фотокадру в режимі реального часу

Методика визначення координат фотокадру полягає у оптимальному виконанні польотного завдання та завдання управління, визначення координатів центрів фотографування та розрахунку координат точок на фотознімку з урахуванням обурень та зміщення центральної проекції фотокадру.

Оптимізація виконання завдання управління полягає у підвищенні ефективності фотознімальних робіт не дивлячись на обурення у вигляді погодних умов та фізичних першкод на маршруті польоту.

Основне завдання управління полягає в перекладі керованого об'єкта з деякого початкового стану (ІПМ) в кінцевий стан (КПМ) по заданій (запланованій, програмній) траєкторії. При реалізації маршрутного

польоту програмної траєкторією є вектор заданих фазових координат. Для забезпечення польоту по заданій траєкторії необхідно безперервно або дискретно управляти його рухом [11].

У задачі управління траєкторією вважаємо, що є об'єкт управління - деяка задана керована динамічна система, що відображається у своєму просторі станів вектором керованих фазових координат ху.

Основна мета управління при вільній маршрутизації полягає у виведенні БПЛА по заданій траєкторії в необхідну область простору за мінімально можливий проміжок часу при гарантованому забезпеченні безпеки польотів і високої точності навігаційних визначень.

З огляду на дії перешкод і збурень точна реалізація польоту по заданій траєкторії, як правило, неможлива. Тому обурена (реальна) траєкторія відрізняється від заданої (запланованої) на розглянутому інтервалі часу. Введемо позначення відхилення керованої траєкторії щодо заданої у вигляді

$$\varepsilon_{\nu} = x_{з,\,n} - x_{y,\,n} \qquad (2.1)$$

Мета синтезу - формування таких управлінь un, для яких керована траєкторія *ху, n* найкращим (оптимальним) чином відтворює (відстежує) задану траєкторію *хз, v*.

Так як процедура оптимізації траєкторії являє собою досить складну задачу, то управління БПЛА можна організувати таким чином, щоб помилка відхилення керованої траєкторії від заданої в будь-який момент часу була мінімально можливою.

Оптимальність управління розуміють в сенсі мінімізації того чи іншого критерію якості. Міра відхилення вибирається в кожному конкретному випадку, і являє собою критерій оптимальності. В інженерній практиці, в сучасній теорії оптимальних систем, широке застосування знаходить узагальнений квадратичний функціонал помилки управління

[10]. В роботі критерії оптимізації траєкторії польоту формулюються у вигляді мінімізації відхилення керованої траєкторії польоту щодо заданої. При вирішенні практичних завдань траекторного управління ПС втрати залежать не від абсолютних значень, а від їх різниці або помилки, при цьому показником якості є узагальнений квадратичний функціонал помилки управління, який стосовно до розглянутого випадку представимо у вигляді:

$$\min M[Acn\ (x_з,\ v,\ x_y,\ v,\ u_n)] \tag{2.2}$$

де *n = 0, N -1* - тимчасовий індекс; *N* - кількість відліків; *Qn* - невід'ємне, певна вагова матриця штрафів на помилки параметрів вектора стану, яка характеризує ступінь важливості відстеження тієї чи іншої компоненти траєкторії; *En* - невід'ємне, певна вагова матриця штрафів яка характеризує витрати на окремі компоненти вектора управління, un - вектор керуючих впливів; *un Î U* - безліч допустимих значень управлінь; cn *(xз, v, xy, v, un)* - функція поточних втрат, яка зростає зі збільшенням відхилення керованої траєкторії щодо заданої і з ростом витрат на управління. Внаслідок невід'ємної визначеності матриць *Qn* і *Pv* квадратичні форми, що входять в функцію втрат - неспадної функції від відхилення (помилки) і відповідно від управління.

Для вирішення завдання оптимального управління в реальних системах найчастіше використовують локальний критерій оптимізації:

$$Jn = M\ \{c_n\ (x_з,\ v,\ x_y,\ v,\ u_n)\}. \tag{2.3}$$

При описі поведінки стохастичних систем та їх управління використовуємо рівняння динаміки зміни фазових координат об'єкта управління і рівняння спостереження за даними фазовими координатами.

Відповідно до моделі траекторного руху (2.1) вектор заданих фазових координат запишемо у вигляді *хз = x, y, z, V, y T*. З метою більшої формалізації наступних викладок з використанням стандартного апарату теорії оптимального управління вводять розширений вектор стану x = (хз, ху) T [12], для якого можна записати різницеве рівняння

$$x_n + 1 = \frac{\Phi_n}{n} + 1_{xn} + B_n u_n + \frac{G_n}{n} + 1_{nx,\,n} \qquad (2.4)$$

де Φn - матриця динаміки системи, Gn - матриця обмежень на шуми системи; nx, n - вектор ДБГШ (Дискретний білий Гауссівський шум) з нульовими математичними очікуваннями і матрцей дисперсій Ψ; *Bn* - вектор коефіцієнтів керуючих впливів системи.

Керуюча система відповідно до деякої стратегією визначає за даними вимірів керуючий вплив. Завдання полягає в тому, щоб знайти стратегію управління, що оптимізує деякий функціонал втрат. У розглянутій постановці потрібно визначити закон керування *uv = uv (ξ1v-1),* оптимальний по локальному критерієм якості [4]

$$un = \arg\min\{Jn\} \qquad (2.5)$$

де показник якості записується як

Jn = Min M {cn (xn, un)}.

При синтезі алгоритму оптимального управління вважаємо, що оцінка стану проводиться на основі обробки спостережень в ІСН.

Для отримання алгоритму управління уявімо показник якості у вигляді

$$Jv = M\{cv(xv, uv)\} = cv(xv, uv(\xi 1v\ -1))\, p\left(xv,\ \xi n1\ -1\right) dxnd\xi 1v - 1 \qquad (2.6)$$

(2.6) усереднення функцій вартості здійснюються за щільностями ймовірностей *p (xv, ξ1v-1)* всіх випадкових величин, а все управління *un1 - 1* вважаються заданими. Уявімо спільну щільність ймовірності, що входить в мінімізуємий функціонал у вигляді твору

$$p\left(Xv,\ \xi 1v-1\right)\ =\ p\left(\frac{xv}{\xi 1v-1}\right)p\ \left(\xi 1v-1\right). \tag{2.7}$$

Особливість даного завдання полягає в залежності щільності ймовірності *p(xv / xv-1, uv)* від управління. При цьому *uv* має бути визначено до кроку і не є випадковим. Сукупність (2.5) - (2.7) являє собою рівняння рішення задачі оптимального керування стохастичними системами по локальному критерію, при цьому основною складовою завдання оптимального управління є завдання оптимальної фільтрації.

Для нелінійних систем наближено справедлива теорема поділу (теорема статистичної еквівалентності), згідно з якою можна окремо синтезувати систему оцінювання параметрів об'єкта і систему оптимального управління [6]. Підставою для цього служить той факт, що при синтезі алгоритмів оптимального оцінювання в інтегрованих системах ПС хорошою збіжності оцінок до істинним фазовим координатах [3].

Оптимально управляти ПС, на який діють випадкові обурення, можна лише оперативно, використовуючи для вироблення керуючого впливу як на апріорну, так і поточну інформацію, яку надає обмірна система і система обробки інформації. Оптимальна система являє собою структуру з негативними зворотніми зв'язками по всіх керованих змінних стану, що свідчить про її високу стійкість. Сигнал управління визначається не станом системи, а її поточної помилкою *εv = xз, n - xy, n* управління.

Структура стохастичної системи траекторного управління з оцінкою стану приведена на Рис. 2.1

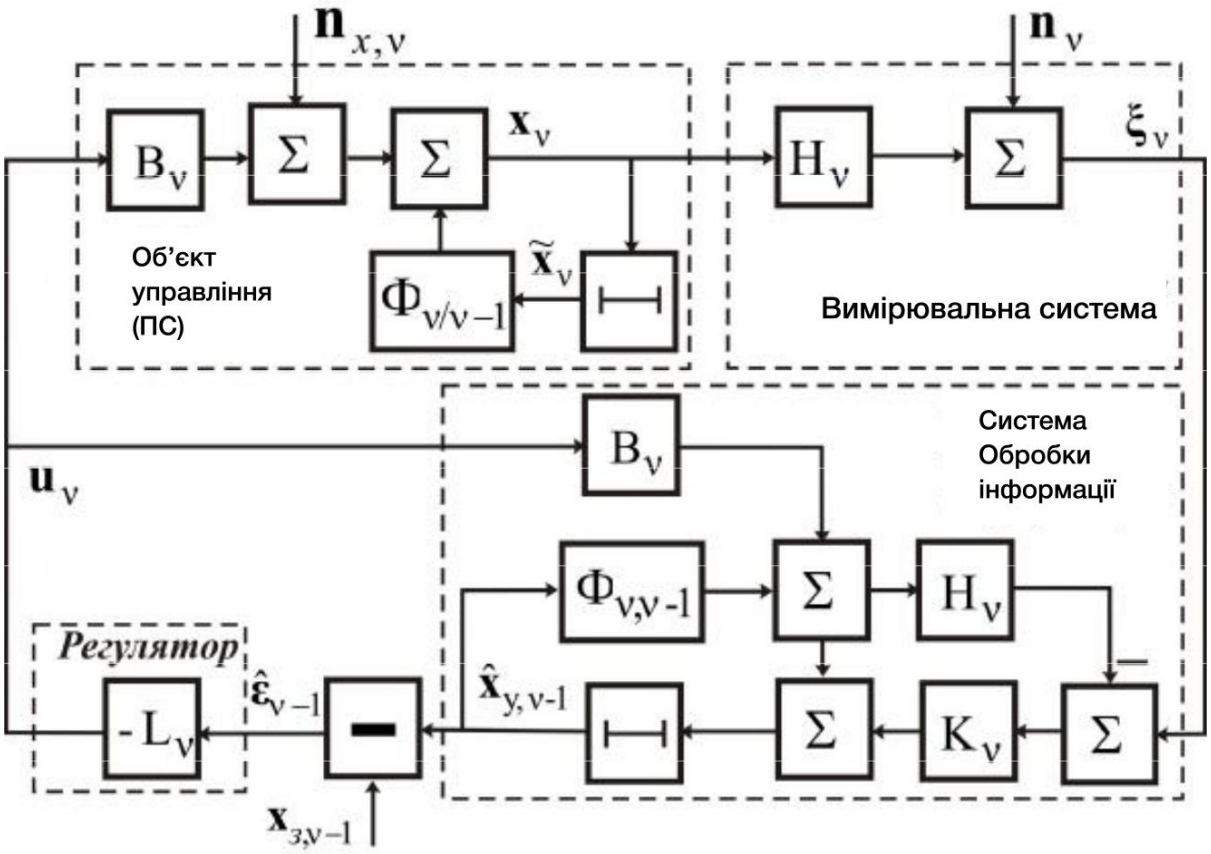


Рис. 2.1 - Структура стохастичної системи траекторного управління з оцінкою стану

2.2. Проектування алгоритму визначення координат фотокадру знятого з безпілотного літального апарату

Перш ніж перейти до песпосередньго наведення координат на фотокадр, слід розглянути процес навігації ПС. Перейдемо до розгляду процессу обробки навігаційних даних.

Практично всі алгоритми комплексування можуть бути отримані з використанням теорії оптимальної фільтрації. В рамках даної теорії існує безліч підходів до синтезу конкретних алгоритмів, фактична відмінність яких полягає у складі використовуваних спостережень (вимірювань) [8].

Методи, які на основі теорії фільтрації широко використовуються при вирішенні завдань синтезу алгоритмів навігаційно-тимчасових визначень в інтегрованих системах навігації. Завдання фільтрації - оцінювання сигналу за спостереженнями, що складається з адитивною суміші сигналу і шуму. Практичному використанню рівнянь фільтрації перешкоджає дві основні проблеми: це вибір математичної моделі вектора стану оцінювання сигналу і вибір відповідних коваріаційних матриць сигналу спостереження. Бажано, щоб модель сигналу описувала його досить повно і в той же час була проста настільки, щоб алгоритми фільтрації піддавався чисельному рішенню. Матриці ковариаций для сигналу і шуму вимірювань також бажано мати такі, щоб вони відповідали реальним процесам. На практиці повний статистичний опис подібних процесів, як правило, важко зробити. Для дослідження показників розбіжності моделі і сигналу використовуються алгоритми аналізу чутливості.

При інтеграції АЗС-М на рівні вторинної обробки найбільш явно проявляється гідність вибору єдиного для обох систем методу вирішення навігаційних завдань. У цьому випадку визначення місця розташування ЗС можна здійснювати на основі наявних вимірів, отриманих від НОТ АЗС-М і видимих на об'єкті НС. Фактично такий підхід дозволяє збільшити число псевдо-віддалемірних спостережень в алгоритмі оцінки координат. Рішення навігаційних завдань в цьому випадку буде здійснюватися шляхом комплексної обробки інформації в єдиному навігаційному обчислювачі. Крім того, очевидно збільшення точності НВО через поліпшення ГФ, т. я. одночасно будуть використовуватися вимірювання від НС і НОТ, що знаходяться вище і нижче щодо ПС.

Для динамічних об'єктів застосування ІСН, які характеризуються більш тривалим автономним (інерціальним) режимом роботи при «збої» первинних вимірах приймальної апаратури ГНСС і наявності зривів стеження за супутниками через затінення прийомної антени елементами

конструкції ВС і відрізняються високою завадостійкістю приймальної апаратури ГНСС, дозволяє забезпечити надійне стеження за навігаційними супутниками при високій динаміці об'єкта і виключити зрив синхронізації при короткочасних завмираннях сигналів ГНСС [7].

Викладена в [3] постановка задачі синтезу комплексної системи навігації з використанням всієї сукупності доступних спостережень передбачає використання стандартного апарату теорії оптимальної фільтрації. Методи калмановської фільтрації активно використовуються при побудові сучасних інтегрованих навігаційних систем різних типів, заснованих на використанні інерційних і супутникових технологій і їх комбінацій показують високу ефективність. У той же час при синтезі алгоритмів комплексування ІСН, можна використовувати модернізований варіант комплексування [4].

Для синтезу алгоритму комплексної вторинної обробки інформації необхідно сформувати вектор спостережень на його вході і визначити склавши оцінку вектора стану, а також врахувати всю апріорну інформацію про систему у вигляді відповідних моделей, що описують стохастичну динаміку вектора стану і його зв'язок з вектором спостереження. При комплексуванні радіотехнічних датчиків (приймача ГНСС і транспондера АЗС-М) з нерадіотехнічними (БІНС і БВ) спостерігається процес для радіотехнічних датчиків, який представляється у вигляді:

$$z_1(T) = S(l,t) + n(t) \qquad (2.8)$$

де *l(t) = cx(t)* - фільтрується процес, який відображається в просторі векторних станів Марковским процесом *x(t); c = 100 ... 0 T; n(t)* - БГШ з кореляційної функцією *M [n (t) nT (t + t)] = N20 d (t), N0* - одностороння спектральна щільність шуму.

Спостереження на виході нерадіотехніческіх датчиків:

$$z_2(T) = l(t) + d(t) \tag{2.9}$$

де *d (t)* - в даному випадку корельований процес *d (t) = bυ (t)*, який в просторі станів відображається Марковским процесом *υ (t).*

Лінійні методи обробки навігаційної інформації вимагають відповідної лінеаризації, як навігаційних вимірювань, так і моделей похибок вимірників, що входять до складу ІСН. Вимірювання, що формуються в ІСН, мають вигляд [7]:

$$z(t) = D(t) = Du(t) - Dp(t) \tag{2.10}$$

де *z (t)* - вектор різницевих вимірювань; *Du (t), D p (t)* - виміряне і розрахункове значення вектора первинних навігаційних параметрів (псевдо і радіальної псевдошвидкості в ГНСС і АЗС-М) які можуть бути представлені у вигляді:

$$Du(t) = F(x(t), t) + n(t) \tag{2.11}$$

$$P(T) = F(\sim x(t), t) = F(x(t) + Dx(t), t) \tag{2.12}$$

де *x(t)* - вектор дійсних значень навігаційних параметрів (координати місця розташування і складові швидкості), приладове значення *~ x (t)* котороговирабативается в БІНС з похибкою *Dx (t) = ~ x (t) - x (t); F (x (t), t)* - відома функція навігаційних параметрів ПС і часу; *n (t)* - вектор похибок вимірювань ГНСС і АЗС-М на етапі первинної обробки навігації.

Вимірювання, що виконуються в ІСН (2.23) можна представити у вигляді

$$z(t) = H[Dx(t), t] + n(t) \tag{2.13}$$

де *H [Dx (t), t]* - нелінійна функція навігаційних параметрів.

Завдання спільної обробки даних в навігаційному обчислювачі ІСН вирішується таким чином. Формуються виміряні значення дальності *D*и, *in* і радіальної швидкості *D* & *i, in* для кожного НС зі складу орбітального угрупування, ПС і наземних станцій взаємодіючих в мережі АЗС-М:

$$Dm, in = [(xin - xmn)2 + (yin - ymn)2 + (zin - zmn)2]\frac{1}{2} \tag{2.14}$$

$$D\,\&\,i, in = Dm + DDin + nD, in \tag{2.15}$$

$$(zin - zmn)(z\,\&\,in - z\,\&\,mn) \tag{2.16}$$

$$D\,\&\,i, in = D\,\&\,m, in + DD\,\&\,in + nD, in \tag{2.17}$$

де *DDin, DD & in* - зміщення відповідно шкали часу і частоти опорного генератора в транспондері АЗС-М щодо даних НСІ; *nD, in, nD &, in* - шуми приймачів ГНСС і транспондера АЗС-М.

На основі даних БІНС і ефемеридної інформації НСІ (НОТ) формуються розрахункові значення дальності і радіальної швидкості для кожного і-того джерела навігаційної інформації:

$$D\,\&\,i, in = D\,\&\,m, in + DD\,\&\,in + nD, in \tag{2.18}$$

$$D_{p,\,in} = \left[\left(Xin - xpn\right)2 + \left(yin - ypn\right)2 + \left(zin - z\,pn\right)2\right]\frac{1}{2} \tag{2.19}$$

Потім для побудови моделі похибок різницевих вимірювань проводиться їх лінеаризація. Для обробки лінеаризованих вимірювань навігаційному процесорі ІСН використовуються математичні моделі, описючі поведінку вектора *Dx (t)* похибок навігаційних вимірювачів, які наведені в розділі 1.

На основі розглянутого модифікованого методу копмплексування можна отримати алгоритм обробки навігаційної інформації в реальному часі. При використанні фільтра Калмана моделі похибок системи і вимірювань можна представити у вигляді системи різницевих рівнянь:

$$D_\nu = \Phi\nu\left(x\nu - 1\right) + B_\nu u\nu - 1 + G_\nu w\nu \qquad (2.20)$$

$$D_\nu = H\nu\left(x\nu\right) + n\nu \qquad (2.21)$$

де *xv* - вектор стану; *Φv (xv-1)* - функція динаміки системи; *Bv* - матриця керуючих впливів; *Gv* - матриця обмежень на шуми системи; *v-1* - вектор управління; *wv* - вектор гауссовских шумів; *zv* - вектор вимірювань (спостережень) системи; *Hv (xv)* - функція вимірювань (спостережень) системи; *nv* - *m*-мірний вектор гауссівських шумів вимірювань (спостережень), що приймається ДБГШ з нульовими математичними очікуваннями матрицею дисперсій *Vv*.

В даний час в задачах навігації при гауссовскій характеристиці шумів *wv* систем і *nv* вимірювань широкого поширення набули алгоритми розширеного (узагальненого) фільтра Калмана, засновані на гауссівській апроксимації апостеріорної щільності при розкладанні в ряд Тейлора функцій динаміки *Φv (xv-1)* і вимірювань *Hv (xv)*. У навігаційних додатках завдання траекторного стеження нелінійності пов'язані з нелінійністю функції в моделі вимірювань. При використанні в якості точок лінеаризації

оцінок *Dхл1 = Dxv-1, Dхл2 = Dxv / v-1* отримаємо алгоритм фільтрації вектора стану на основі розширеного (узагальненого) фільтра Калмана [7]:

$$\frac{Dxv}{v-1} = \Phi v\ (Dхл_1) + \Phi v\ (Dхл_1)\ (Dxv-1 - Dхл_1),\ DxT \tag{2.22}$$

$$Dxv = \frac{Dxv}{v-1} + Kv(Dхл_1,\ Dхл_2)\left[zv - Hv(Dхл_2) - Hv(Dхл_2)\left(\frac{Dxv}{v\ -1\ -\ Dхл_2}\right)\right],\ DxTv \tag{2.23}$$

де матриці *Pv / v-1 (Dхл1), Pv (Dхл1, Dхл2*) і матричний коефіцієнт посилення *Kv (Dхл1, Dхл2)* будуть визначатися відповідно до виражень для лінійної фільтрації, тобто, при реалізації даного підходу за даними з виходу ІСН *λn* і оцінений-ним значенням змінних вектора стану *D*xn визначається оцінка вектора інформаційного процесу, що містить просторові координати ПС по формулі

$$xn = \lambda n - Dxn \tag{2.24}$$

потім використовується в ЗПП для формування оптимальної стратегії управлінні траєкторією [5].

З практичної точки зору в даний час найбільш доцільним є застосування децентралізованого алгоритму (ДА) з гнучкою логікою. Ієрархія передбачає зміну показників якості ПС в часі, які необхідно включати до складу повідомлень якими обмінюються взаємодіючі об'єкти. Склад і довжина повідомлень при автоматичному залежному спостереженні наведені в [6]. Там же зазначені умови передачі повідомлень, з яких випливає, що тільки координати і показник якості їх визначення рекомендується передавати в кожному повідомленні, а всі інші дані з борту ПС - за запитом.

Концепція ТАК передбачає некорреліроване похибки оцінювання з помилками оцінювання БВП інших об'єктів [5]. Матриця *Pv* розраховується на основі прийнятих від НОТ показників якості місця розташування і часу за наявною таблиці відповідності показників якості середньо-квадратичних похибок визначення навігаційних параметрів. Використання даного підходу, природно, призводить до збільшення дисперсії еквівалентного шуму. Однак, таке збільшення є одним з відомих способів підвищення стабільності фільтра Калмана і згідно з результатами [5], дійсні помилки такого фільтра не перевищуватимуть розрахункових.

Тепер, коли ми маємо навігаційні дані повітряного судна і його координати у просторі, ми можемо перейти до процесу визначення координат центрів фотогрфування та відновлення, спотвореної в наслідок технічних та природних обурень, центральної проекції фотокадру для більш точного визначення координат точок на фотографії.

Для початку визначимо дані, якими будемо оперувати в процесі.

В якості вхідних даних будемо використовувати лог фотозйомки з БПЛА. Ці данні є записами положення та властивостей орієнтації повітряного судна під час подачі сигналу для фотографування.

Визначимо ці показники:

- *latitude / longtitude* – є координатами широти і довготи повітряного судна. Для спрощення процесу оперування цими даними, ми представимо їх в якості цілочисленної кількості десяти мільйонних часток градуса в північній півкулі та східній півсфері.
- *altitude mean sea level* – рівень підйому повітряного судна над рівнем моря.
- *altitude relative* – рівень підйому повітряного судна щодо стартової установки (будемо вважати, що БПЛА запускається з поверхні землі та представимо цей показник, як висоту повітряного судна щодо земної поверхні).

- *roll* – показник крену повітряного судна. Якщо значення більше нуля, то ПС поветраеться за годинниковою стрілкою, якщо менше – то проти.
- *pitch* – тангаж. Показник, який визначає нахил носової частини повітряного судна вверх та вниз. Якщо значення більше за нуль, то ніс літального апарату направлений вгору, якщо менше нуля – то вниз.
- *yaw* – курс руху повітряного судна. Якщо значення дорівнює нулю, то літальний апарат прямує точно на північ.

Також, в розрахунках будуть використовуватися параметри камери, такі як:

- focal length – фокусна відстань. Це відстань між фокусом об’єктиву і його оптичним центром[15].
- Широта і висота матриці знімального апарату

Тепер, можемо побудувати наступну математичну модель.

Першим кроком розрахуемо кути огляду площини *a* та *b* за відомими нам параметрами камери.

$$a = \arccos(1 - \frac{2c^2}{c^2 + 4F^2}) \tag{2.25}$$

$$b = \arccos(1 - \frac{2d^2}{d^2 + 4F^2}) \tag{2.26}$$

Де *c* – це широта матриці, *d* – висота матриці, *F* – фокусна відстань об’єктиву. Далі буде розрахунок висоти трикутнику поля бачення літального апарату

$$h1 = H * tg(\frac{a}{2} + x) \tag{2.27}$$

$$h1 = H * tg(\frac{a}{2} - x) \tag{2.28}$$

$$h1 = H * tg(\frac{b}{2} + y) \tag{2.29}$$

$$h1 = H * tg(\frac{b}{2} - y) \tag{2.30}$$

Де x – показник тангажу, y – показник крену, a та b – кути огляду фотокамери а Н – висота повітряного судна.

Наступним кроком, ми розрахуємо довжину гіпотинузи трикутників поля бачення БПЛА за формулою.

$$l_1 = \frac{h1}{\sin(\frac{a}{2} - x)} \tag{2.31}$$

$$l_2 = \frac{h2}{\sin(\frac{a}{2} + x)} \tag{2.32}$$

$$l_3 = \frac{h3}{\sin(\frac{a}{2} + y)} \tag{2.33}$$

$$l_4 = \frac{h4}{\sin(\frac{a}{2} - y)} \tag{2.34}$$

Де h – висота поля трикунтика поля бачення.

Тепер розрахуємо сторону трикутника, що є лінією поверхні землі.

$$d_1 = l_1 tg\frac{b}{2} \tag{2.35}$$

$$d_2 = l_2 tg\frac{b}{2} \tag{2.36}$$

$$d_3 = l_3 tg\frac{a}{2} \tag{2.37}$$

$$d_4 = l_4 tg\frac{a}{2} \tag{2.38}$$

Де d – сторона трикутника, що є віссю x, l – гіпотенуза трикутника.

Далі розрахуємо довжини векторів.

$$|\vec{c_1}| = \sqrt{(d_1)^2 + (h_1)^2} \tag{2.39}$$

$$|\vec{c_2}| = \sqrt{(d_2)^2 + (h_2)^2} \tag{2.40}$$

$$|\vec{c_3}| = \sqrt{(d_3)^2 + (h_3)^2} \tag{2.41}$$

$$|\vec{c_4}| = \sqrt{(d_4)^2 + (h_4)^2} \tag{2.42}$$

Де *c* – є вектором. Тепер розрахуємо ідеальне розміщення вектора.

$$a_3 = 2Htg\frac{a}{2} \tag{2.43}$$

$$b_3 = 2Htg\frac{b}{2} \tag{2.44}$$

$$|\vec{a_i}| = \frac{1}{2}\sqrt{(a_3)^2 + (b_3)^2} \tag{2.45}$$

Далі знайдемо від'ємний вектор, який необхідний для визначення реального вектору для кожної площини.

$$((\overline{a_{1,2,3,4}} - \overline{a_{ИД}})\eta) = arcctg(\frac{\overline{a_{1,2,3,4}}\cos < (\overline{a_{1,2,3,4}} : \eta) - \overline{a_{ИД}}\cos < (\overline{a_{ИД}} : \eta)}{\overline{a_{1,2,3,4}}\sin < (\overline{a_{1,2,3,4}} : \eta) - \overline{a_{ИД}}\sin < (\overline{a_{ИД}} : \eta)}) \tag{2.46}$$

Тепер знайдемо вектор зміщення для кожної площини.

$$\overline{a_z} = \overline{a_i} + \overline{a_3} \tag{2.47}$$

Далі обчислимо координати крайніх точок за широтою та довготою.

$$l_{latitude} = \frac{1}{2}R_3(\frac{1}{180*60*60}) \tag{2.48}$$

За широтою

$$l_{longtitude} = \frac{2\pi R_3 \cos a}{360*60*60} \tag{2.49}$$

За довготою

Звідси ми отримуємо, що:

Координати по широті точки:

$$latitude = \frac{a_z \sin z}{l_{latitude}} \tag{2.50}$$

Координати по довготі точки:

$$longtitude = \frac{a_z \sin z}{l_{longtitude}} \tag{2.51}$$

Зараз ми розрахували всі, необхідні, проміжні значення для всіх крайніх точок фотокадру. Далі ми можемо приступати до програмної реалізації алгоритму.

2.3. Вибір та обґрунтування засобів реалізації системи

При виборі засобів реалізації системи, ми будемо розглядати дані проведеного експерименту[6].

Спочатку визначимося з типом безпілотного літального апарату. На даний момент використовуються переважно літальні апарати вертолітного та літакового типів (рис 2.3).

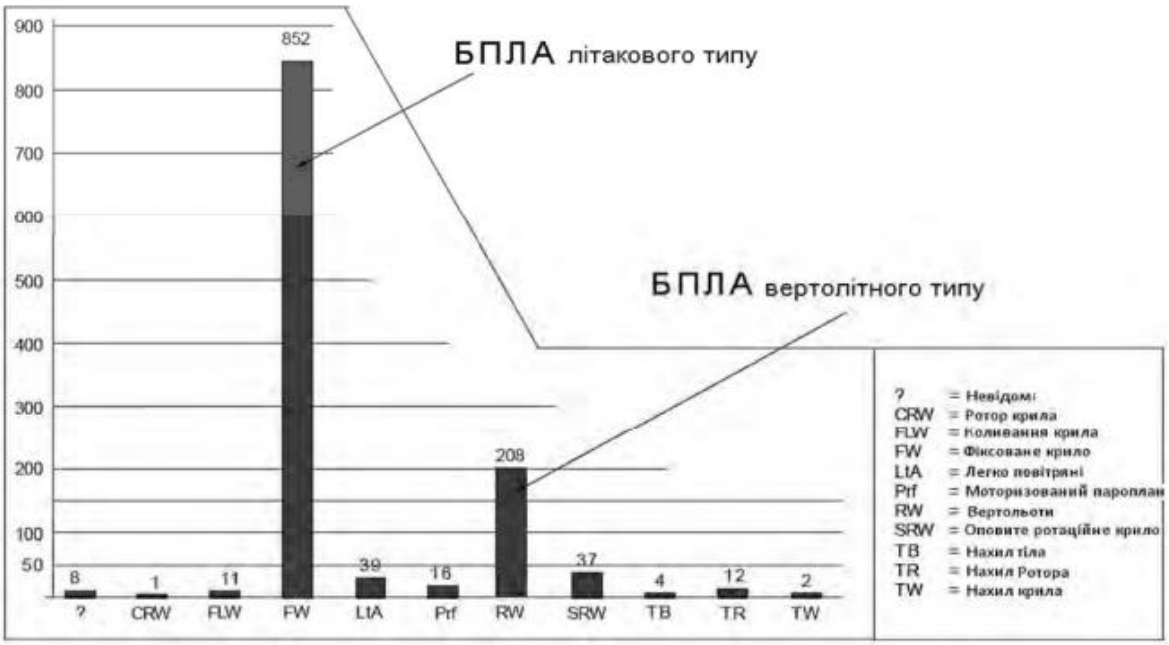


Рис 2.3 – Співвідношення кількості БПЛА вертолітного та літакового типу відносно всіх інших типів за даними UVS International

Впродовж першого етапу експерименту аерофотозйомка виконувалася з борту БПЛА “Птах” сконструйованого ФАКС НТУУ-КПІ(рис. 2.4).

Рис 2.4 – БПЛА “ФАКС-4 ПТАХ”

Технічні характеристики літального апарату:

Злітна маса	65 кг
Розмах крила	5 м
Довжина	2.5 м
Швидкість польоту	80-150 км/год
Маса корисного навантаження	25 кг
Час польоту з вантажем 25 кг	4 год
Маса планера	32 кг
Маса палива	5-10 кг
Двигун двоциліндровий	15 к.с.
Злітна швидкість	60 км/год
Посадкова швидкість	50 км/год
Злітно-посадкова смуг	50-70 м
Максимальна швидкість набору висоти	6.5 м/с
Аеродинамічна якість, 95 км/год	15

Проаналізувавши результати, було виявлено ряд недоліків:

1) Не точне дотримання швидкості
2) Нестабільна робота системи керування літального апарату на відстанях понад 700 метрів
3) Рівень вібрації літака приводив до змазування зображень на фотокадрах.

З урахуванням цих проблем, було спроектовано новий безпілотний літак малих розмірів та з електричним двигуном – “Пегас”(рис. 2.5).

Рис 2.4 – БПЛА “ФАКС-5 ПЕГАС”

Злітна маса	12 кг
Розмах крила	2.4 м
Довжина	1.7 м
Швидкість польоту	50-110 км/год
Маса корисного навантаження	5-7.5 кг
Час польоту з вантажем 25 кг	4 год
Маса планера	8 кг
Двигун електричний	2-3.5 кВт
Злітна швидкість	40 км/год
Посадкова швидкість	40 км/год
Злітно-посадкова смуг	25 м
Максимальна швидкість набору висоти	4 м/с
Аеродинамічна якість, 95 км/год	11
Радіус віражу	30 м

В якості знімального пристрою була використана камера Canon EOS 450D з фокусною відстанню в 18 мм, кроп-фактором 1.62 та розмірами матриці за висотою 22 мм та за шириною 14 мм . Заплановані перекриття в експеременті становили 80% горизонтально перекриття та 40% вертикального. Висота літака під час знімальних робіт становила 300

м, шляхова швидкість – 60 км/год. З урахуванням висоти та фокусної відстані оптимальним масштабом знімання був прийнятий 1:17000.

3. РЕАЛІЗАЦІЯ СИСТЕМИ ВИЗНАЧЕННЯ КООРДИНАТ ФОТОКАДРУ ЗНЯТОГО З БЕЗПІЛОТНОГО ЛІТАЛЬНОГО АПАРАТУ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

3.1. Розробка системи

Програмна реалізація системи, реалізована мовою програмування Java та включає в себе програмний рушій та модулі, які представляють собою систему оброки (“парсингу”) та підготовки до використання вхідних файлів, таких як лог аерофотозйомки, який має вигляд таблиці з розширенням .csv, та дані якого мають бути підготовлення та сумісні із мовою програмування, також є модуль виводу даних, який створює файл розширення .txt, що містить в собі кінцевий результат розрахунків координат. Програмний рушій, в свою чергу, представляє собою набір моделей, в яких оголошуються змінні якими оперують в коді та їх тип, та утиліт в яких прописані математичні перетворення с цими значеннями. При розробці, для спрощення та зменшення коду, були використані бібліотеки Project Lombok.

Розпочнемо з оголошення ввідних змінних. Для їх оголошення створено окремий клас Variables.

```
package model;

import lombok.Builder;
import lombok.Value;

@Value
```

```
@Builder
public class Variables {
        double latitude;
        double longtitude;
        int h;

        double roll;
        double pitch;
        double yaw;

}
```

В цьому класі оголошено змінну *latitude* – координати по широті, *longtitude* – координати по довготі, *h* – висота, *roll, pitch, yaw* – крен, тангаж та курс відповідно.

Далі розглянемо класс з константами – параметрами камери, які не змінються в процесі.

```
package model;

public class ConstValues {
        public static final double frame_h = 0.22 / 1.62;
        public static final double frame_w = 0.14 / 1.62;
        public static final double F = 0.18;

}
```

frame_h, frame_w – висота та ширина проекції знімку відповідно, є відношенням висоти та ширини матриці камери до кроп-фактору, *F* – фокусна відстань.

Тепер варто звернути увагу на модулі обробки та виводу даних, так як вони беруть безпосередню участь в автоматизації процесу.

Спочатку розглянемо модуль, який зчитує дані з логу аерофотозйомки і перетворює їх в ті типи даних, якими ми можемо оперувати в програмному коді.

```
package module;

import java.io.*;

import model.Variables;
```

```
public class FileParser {

	private String input;

	public FileParser(String input) {
		this.input = input;
	}

	private String getRaw(int i) throws IOException {
		int line = 0;
		BufferedReader BufferedReader = new BufferedReader(new
FileReader(input));
		while (line < i) {
			BufferedReader.readLine();
			line++;
		}
		return BufferedReader.readLine();
	}

	private String[] splitRaw(String raw) {
		return raw.split(",");
	}

	public Variables getParsedData(int id) throws IOException {
		String raw = getRaw(id);
		String[] data = splitRaw(raw);
		return
Variables.builder().latitude(Double.parseDouble(data[12]) / 10000000)
				.longtitude(Double.parseDouble(data[14]) /
10000000).h((int) (Double.parseDouble(data[18])))

	.roll(Double.parseDouble(data[20])).pitch(Double.parseDouble(data[2
2]))
				.yaw(Double.parseDouble(data[24])).build();
	}

	public int getCount() throws IOException {
		int count = 0;
		BufferedReader BufferedReader = new BufferedReader(new
FileReader(input));
		while (BufferedReader.readLine() != null) {
			count++;
		}
		return count;
	}
}
```

Даний “парсер” призначений для синтаксичного аналізу даних з таблиці з розширенням *.csv*. Так як процес аерофотозйоки передбачає фотографування земної поверхні з малими інтервалами у часі, то слід очікувати велику кількість фотографій за один прольот маршруту. Звідси виходить и велика кількість записів в лог зйомки, тому парсер має

автоматично зчитувати дані кожного фотокадру з таблиці. Після аналізу та перетворення значень за таблиці в типи даних мови програмування вони передаються в оголошені змінні які були створені для них. Дані вписані в колонки таблиці і для їх считування для кожної зміної номер колонки з її даними був заданий виходячи із порядку цих колонок, тому при зміні формату логу необхідно вручну вказати порядковий номер колонки для кожної змінної вхідних даних.

Тепер розглянемо безпосередню програмну реалізацію математичної моделі.

```
package utilities;

import static java.lang.Math.*;
import static model.ConstValues.*;

import model.Variables;

public class Calculations {

	private double getAlphaPitch(double height) {
		return height * sqrt(pow(F, 2) + pow(frame_w, 2) / 4);
	}

	private double getAlphaRoll(double height) {
		return height * sqrt(pow(F, 2) + pow(frame_h, 2) / 4);
	}

	private double getBetaPitch() {
		return atan(frame_w / (2 * F));
	}

	private double getBetaRoll() {
		return atan(frame_h / (2 * F));
	}

	private double x1_calculation(double x_center, double height,
double pitch, double roll, double yaw) {
		double AlphaPitch = getAlphaPitch(height);
		double BetaPitch = getBetaPitch();
		double BetaRoll = getBetaRoll();

		double x_pitch = pitch > 0 ? AlphaPitch / cos(pitch +
BetaPitch) : AlphaPitch / cos(pitch - BetaPitch);
		double x_roll = roll > 0 ? height * (sin(BetaRoll) /
pow(cos(roll), 2))
				: height / (cos(roll) * cos(BetaRoll) +
sin(roll) * sin(BetaRoll));

		return (x_center + x_pitch + x_roll);
```

```
        }

        private double x2_calculation(double x_center, double height,
double pitch, double roll, double yaw) {

                double AlphaPitch = getAlphaPitch(height);
                double BetaPitch = getBetaPitch();
                double BetaRoll = getBetaRoll();

                double x_pitch = pitch > 0 ? AlphaPitch / cos(pitch -
BetaPitch) : AlphaPitch / cos(pitch + BetaPitch);
                double x_roll = roll > 0 ? height * (sin(BetaRoll) /
pow(cos(roll), 2))
                                : height / (cos(roll) * cos(BetaRoll) +
sin(roll) * sin(BetaRoll));

                return (x_center + x_pitch + x_roll);
        }

        private double x3_calculation(double x_center, double height,
double pitch, double roll, double yaw) {

                double AlphaPitch = getAlphaPitch(height);
                double BetaPitch = getBetaPitch();
                double BetaRoll = getBetaRoll();

                double x_pitch = pitch > 0 ? -AlphaPitch / cos(pitch -
BetaPitch) : -AlphaPitch / cos(pitch + BetaPitch);
                double x_roll = roll > 0 ? -height / (cos(roll) *
cos(BetaRoll) + sin(roll) * sin(BetaRoll))
                                : -height * (sin(BetaRoll) / pow(cos(roll),
2));

                return (x_center + x_pitch + x_roll);
        }

        private double x4_calculation(double x_center, double height,
double pitch, double roll, double yaw) {

                double AlphaPitch = getAlphaPitch(height);
                double BetaPitch = getBetaPitch();
                double BetaRoll = getBetaRoll();

                double x_pitch = pitch > 0 ? -AlphaPitch / cos(pitch +
BetaPitch) : -AlphaPitch / cos(pitch - BetaPitch);
                double x_roll = roll > 0 ? -height / (cos(roll) *
cos(BetaRoll) + sin(roll) * sin(BetaRoll))
                                : -height * (sin(BetaRoll) / pow(cos(roll),
2));

                return (x_center + x_pitch + x_roll);
        }

        private double y1_calculation(double y_center, double height,
double pitch, double roll, double yaw) {

                double AlphaRoll = getAlphaRoll(height);
                double BetaPitch = getBetaPitch();
                double BetaRoll = getBetaRoll();
```

```
                double y_pitch = pitch > 0 ? height * (sin(BetaPitch) /
pow(cos(pitch), 2))
                                : height / (cos(pitch) * cos(BetaPitch) +
sin(pitch) * sin(BetaPitch));
                double y_roll = roll > 0 ? AlphaRoll / cos(roll + BetaRoll)
: AlphaRoll / cos(roll - BetaRoll);

                return (y_center + y_pitch + y_roll);
        }

        private double y2_calculation(double y_center, double height,
double pitch, double roll, double yaw) {

                double AlphaRoll = getAlphaRoll(height);
                double BetaPitch = getBetaPitch();
                double BetaRoll = getBetaRoll();

                double y_pitch = -height / (cos(pitch) * cos(BetaPitch) +
sin(pitch) * sin(BetaPitch));
                double y_roll = roll > 0 ? -AlphaRoll / cos(roll +
BetaRoll) : -AlphaRoll / cos(roll - BetaRoll);

                return (y_center + y_pitch + y_roll);
        }

        private double y3_calculation(double y_center, double height,
double pitch, double roll, double yaw) {

                double AlphaRoll = getAlphaRoll(height);
                double BetaPitch = getBetaPitch();
                double BetaRoll = getBetaRoll();

                double y_pitch = -height / (cos(pitch) * cos(BetaPitch) +
sin(pitch) * sin(BetaPitch));
                double y_roll = roll > 0 ? -AlphaRoll / cos(roll -
BetaRoll) : -AlphaRoll / cos(roll + BetaRoll);

                return (y_center + y_pitch + y_roll);
        }

        private double y4_calculation(double y_center, double height,
double pitch, double roll, double yaw) {

                double AlphaRoll = getAlphaRoll(height);
                double BetaPitch = getBetaPitch();
                double BetaRoll = getBetaRoll();

                double y_pitch = pitch > 0 ? height * (sin(BetaPitch) /
pow(cos(pitch), 2))
                                : height / (cos(pitch) * cos(BetaPitch) +
sin(pitch) * sin(BetaPitch));
                double y_roll = roll > 0 ? AlphaRoll / cos(roll - BetaRoll)
: AlphaRoll / cos(roll + BetaRoll);

                return (y_center + y_pitch + y_roll);
        }

        public String getResult(Variables var) {
```

```
                double pitch_ = var.getPitch();
                double roll_ = var.getRoll();
                double yaw_ = var.getYaw();
                double height = var.getH();
                double x0 = var.getLatitude();
                double y0 = var.getLongtitude();
                double pitch = pitch_ / 180 * PI;
                double roll = roll_ / 180 * PI;
                double yaw = yaw_ / 180 * PI;
                double x_center = height * tan(roll);
                double y_center = height * tan(pitch);
                double x1 = x1_calculation(x_center, height, pitch, roll,
yaw);
                double x2 = x2_calculation(x_center, height, pitch, roll,
yaw);
                double x3 = x3_calculation(x_center, height, pitch, roll,
yaw);
                double x4 = x4_calculation(x_center, height, pitch, roll,
yaw);
                double y1 = y1_calculation(y_center, height, pitch, roll,
yaw);
                double y2 = y2_calculation(y_center, height, pitch, roll,
yaw);
                double y3 = y3_calculation(y_center, height, pitch, roll,
yaw);
                double y4 = y4_calculation(y_center, height, pitch, roll,
yaw);
                double x_center_yaw = x_center * cos(yaw) + y_center *
sin(yaw);
                double y_center_yaw = -x_center * sin(yaw) + y_center *
cos(yaw);
                double x1_yaw = x1 * cos(yaw) + y1 * sin(yaw);
                double x2_yaw = x2 * cos(yaw) + y2 * sin(yaw);
                double x3_yaw = x3 * cos(yaw) + y3 * sin(yaw);
                double x4_yaw = x4 * cos(yaw) + y4 * sin(yaw);
                double y1_yaw = -x1 * sin(yaw) + y1 * cos(yaw);
                double y2_yaw = -x2 * sin(yaw) + y2 * cos(yaw);
                double y3_yaw = -x3 * sin(yaw) + y3 * cos(yaw);
                double y4_yaw = -x4 * sin(yaw) + y4 * cos(yaw);
                double x_center_geo = ValuesConverter.Meters_Longtitude(0,
x_center_yaw, x0) + x0;
                double y_center_geo = ValuesConverter.Meters_Longtitude(0,
y_center_yaw, y0) + y0;
                double x1_coord = ValuesConverter.Meters_Longtitude(0,
x1_yaw, x0) + x0;
                double x2_coord = ValuesConverter.Meters_Longtitude(0,
x2_yaw, x0) + x0;
                double x3_coord = ValuesConverter.Meters_Longtitude(0,
x3_yaw, x0) + x0;
                double x4_coord = ValuesConverter.Meters_Longtitude(0,
x4_yaw, x0) + x0;
                double y1_coord = ValuesConverter.Meters_Latitude(0,
y1_yaw) + y0;
                double y2_coord = ValuesConverter.Meters_Latitude(0,
y2_yaw) + y0;
                double y3_coord = ValuesConverter.Meters_Latitude(0,
y3_yaw) + y0;
                double y4_coord = ValuesConverter.Meters_Latitude(0,
y4_yaw) + y0;
```

```
                String result =
String.format("%s,%s\n%s,%s\n%s,%s\n%s,%s\n%s,%s\n", y_center_geo,
x_center_geo, y1_coord,
                                    x1_coord, y2_coord, x2_coord, y3_coord,
x3_coord, y4_coord, x4_coord);
                System.out.println(result);
                return result;
        }
}
```

В цьому сегменті відбуваються обчислення представлені в математичній моделі. Результатом обчислень є координати 5-ти точок – центральної та чотирьох крайніх точок фотокадру, які конвертуються в формат десятичних градусів та записуються в *String* для подальшого запису в файл виведення.

Код конвертації даних координат з метрів в десятичні градуси.

```
package utilities;

import static java.lang.Math.cos;

public class ValuesConverter {
    public static double Meters_Latitude(double m1, double m2) {
        double deegres_in_meters = 1 / 111134.861111;
        return (m2 - m1) *deegres_in_meters;
    }



        public static double Meters_Longtitude(double m1, double m2, double
longe) {
        double deegres_in_meters = 1 / (111321.377778 * cos(longe));
        return (m2 - m1) * deegres_in_meters;
    }
}
```

Тепер розглянемо модуль запису даних в вихідний файл.

```
package module;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class FileOutput {
        public void WriteOutputToFile(String log) throws IOException {
```

```
                FileWriter fwriter = new FileWriter("output.txt", true);
                BufferedWriter bwriter = new BufferedWriter(fwriter);
                try (PrintWriter out = new PrintWriter(bwriter)) {
                        out.println(log);
                }
        }
}
```

Представлений класс включає в себе метод створення файлу та запису в нього даних.

```
package engine;

import utilities.Calculations;

import java.io.IOException;

import model.ConstValues;
import model.Variables;
import module.FileParser;
import module.FileOutput;
```

Клас *Controller* бере на себе роль запуску процесу зчитування даних з логу та запису вихідних даних у файл.

```
public class Controller {
        Variables var;
        FileParser parser = new FileParser("input.csv");
        FileOutput fileOutput = new FileOutput();

        public void OnePhotoStart(int id) throws IOException {
                var = parser.getParsedVars(id);
                new Calculations().getResult(var);
        }

        public void SetPhotoStart() throws IOException {



        int count = parser.getCount();
                for (int id = 1; id < count; id++) {
                        var = parser.getParsedVars(id);
                        String result_out = new
Calculations().getResult(var);
                        fileOutput.WriteOutputToFile(result_out);
                }
        }
}
```

В класі *Controller* передбачені два методи. Перший метод обробляє дані та записує дані лише однієї фотографії, другий автоматизовану обробку великої кількості фотографій.

В класі *Main* представлені об`єкти класу *Controller,* які відповідають за те в якому режимі буде праювати программа.

```
package main;

import java.io.IOException;

import engine.Controller;

public class Main {

	public static void main(String[] args) throws IOException {
		Controller controller = new Controller();
		// controller.OnePhoto(1);
		controller.MultiplePhotos();
	}
}
```

3.2. Тестування системи

Проведемо тестування системи на основі фотографування населеного пункту Оверята, Пермського краю (рис3.1). Знімок був зроблений камерою Sony DSC-RX100 з фокусною відстанню 10 мм, висотою та шириною матриці – 13.2 та 8.8 мм відповідно та кроп-фактором 2.7.

Рис 3.1 – Населний пункт

Координати літального апарату під час зйомки:

latitude - 58.10635234952969

longtitude - 55.85532494234591

altitude mean sea level – 437

relative altitude – 287

roll – 0.33

pitch – 0.2

yaw – 125

На виході ми отримал наступні координати(Рис 3.2)

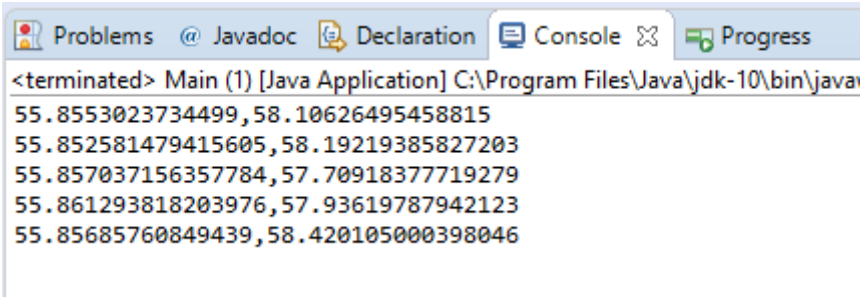


Рис 3.2 – координати центральної та крайніх точок

Їх можна первірити за допогою Google Maps вписавши туди першу строку координат, що є координатами центральної точки (Рис 3.3)

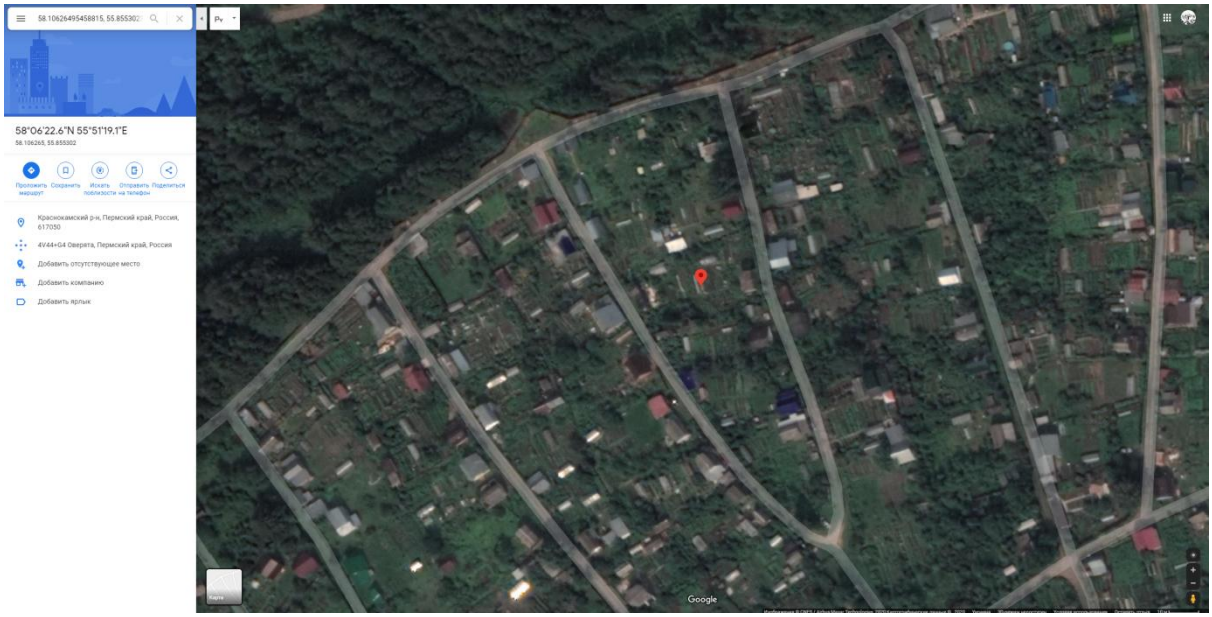


Рис 3.3 – Первірка результатів випробування

Як ми бачимо, результати можно назвати задовільними.

3.3. Інструкція користувача

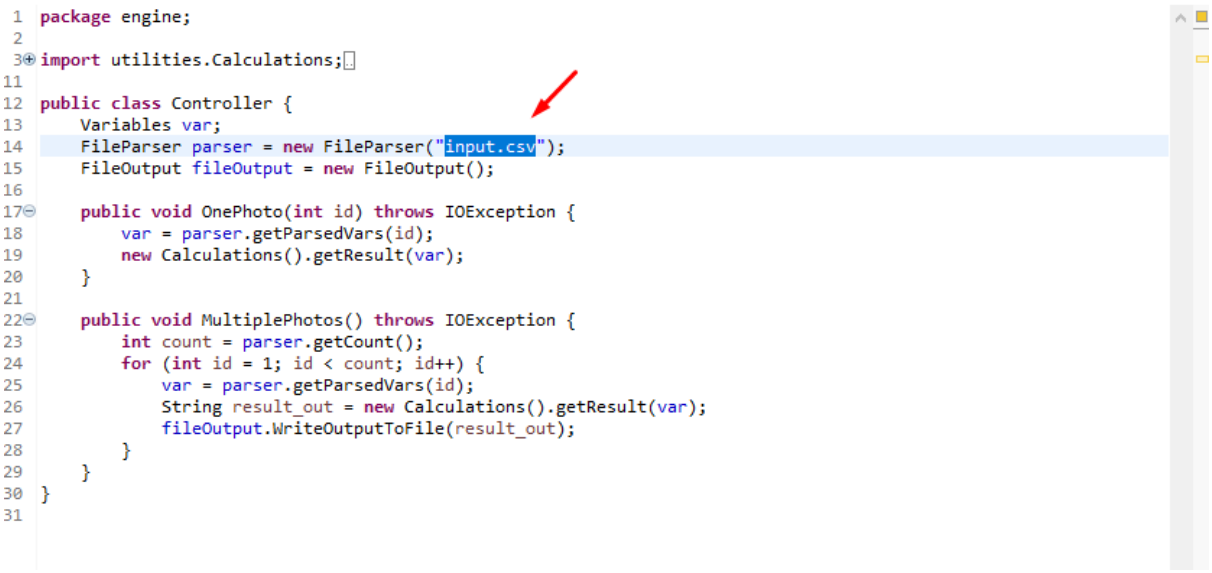
Для використання програми слід підготовити файл з вхідними даними, лог аерофотозйомки в розширені *.csv*. В файлі мають бути дані, які були перераховані вище.

L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
	lat		lon		alt_msl		alt_rel		roll		pitch		yaw
lat	474317996	lng	379570196	alt_msl	427.49	alt_rel	348.78	roll	1.23	pitch	0.11	yaw	109.32
lat	474315877	lng	379583163	alt_msl	426.04	alt_rel	347.33	roll	5.33	pitch	1.84	yaw	103.2
lat	474313066	lng	379595871	alt_msl	424.9	alt_rel	346.19	roll	0.88	pitch	2.38	yaw	106.89
lat	474310386	lng	379608719	alt_msl	425.65	alt_rel	346.94	roll	1.65	pitch	0.72	yaw	105.88
lat	474307609	lng	379621429	alt_msl	410.61	alt_rel	331.9	roll	-1.16	pitch	-11.31	yaw	112.97
lat	474304771	lng	379634223	alt_msl	385.06	alt_rel	306.35	roll	-0.8	pitch	-2.35	yaw	109.37
lat	474302520	lng	379647121	alt_msl	371.96	alt_rel	293.25	roll	1.61	pitch	1.87	yaw	106.8
lat	474299956	lng	379660067	alt_msl	375.68	alt_rel	296.97	roll	0.01	pitch	3.17	yaw	105.94
lat	474297443	lng	379672917	alt_msl	378.77	alt_rel	300.06	roll	1.87	pitch	1.52	yaw	106.13
lat	474294678	lng	379685556	alt_msl	380.84	alt_rel	302.13	roll	-0.56	pitch	-1.03	yaw	107.5
lat	474292212	lng	379698521	alt_msl	377.67	alt_rel	298.96	roll	0.37	pitch	0.44	yaw	106.87
lat	474289463	lng	379711589	alt_msl	377.73	alt_rel	299.02	roll	1.24	pitch	1.12	yaw	107.43
lat	474286880	lng	379724518	alt_msl	378.88	alt_rel	300.17	roll	2.07	pitch	1.04	yaw	104.47
lat	474284243	lng	379737366	alt_msl	379.68	alt_rel	300.97	roll	0.18	pitch	-0.26	yaw	107.07
lat	474281520	lng	379750371	alt_msl	380.79	alt_rel	302.08	roll	-1.13	pitch	-0.8	yaw	109.41
lat	474279009	lng	379763403	alt_msl	379.56	alt_rel	300.85	roll	0.36	pitch	-1.48	yaw	108.82
lat	474276373	lng	379776700	alt_msl	378.42	alt_rel	299.71	roll	3.43	pitch	-0.55	yaw	107.67
lat	474273642	lng	379789760	alt_msl	379.29	alt_rel	300.58	roll	3.34	pitch	-0.47	yaw	105.72
lat	474271010	lng	379802681	alt_msl	378.26	alt_rel	299.55	roll	2.54	pitch	0.84	yaw	102.26

Рис 3.4 – Вид логу аерофотозйоки

Координати за широтою та довготою мають бути подані в десятичному вигляді.

Після підготовки вхідного файлу можна вказати його розташування в програмному коді, в класі *Controller* (Рис 3.2)

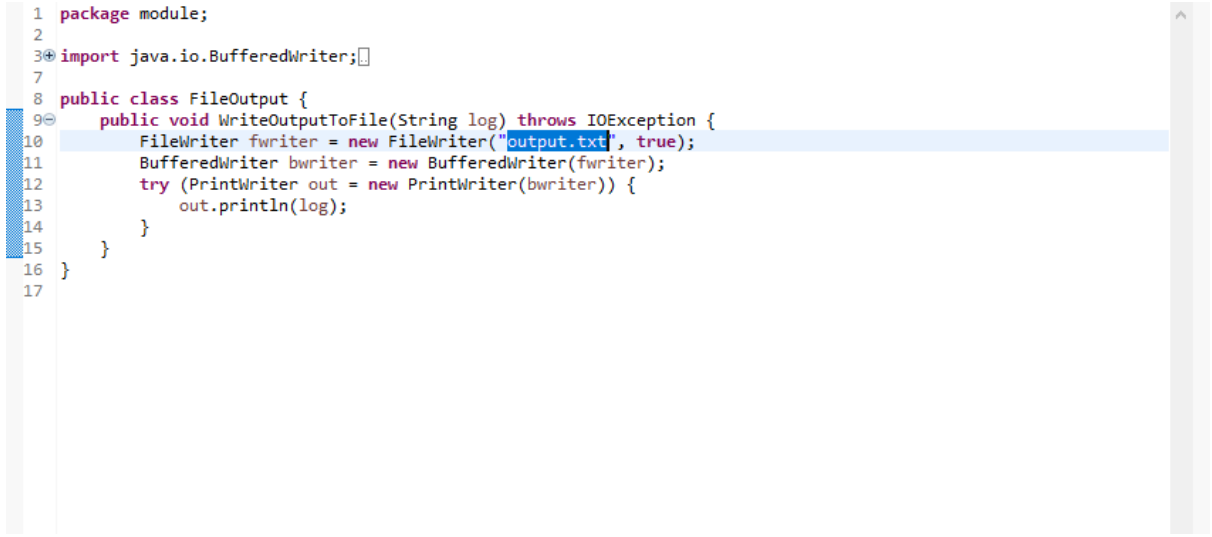


```
package engine;

import utilities.Calculations;

public class Controller {
    Variables var;
    FileParser parser = new FileParser("input.csv");
    FileOutput fileOutput = new FileOutput();

    public void OnePhoto(int id) throws IOException {
        var = parser.getParsedVars(id);
        new Calculations().getResult(var);
    }

    public void MultiplePhotos() throws IOException {
        int count = parser.getCount();
        for (int id = 1; id < count; id++) {
            var = parser.getParsedVars(id);
            String result_out = new Calculations().getResult(var);
            fileOutput.WriteOutputToFile(result_out);
        }
    }
}
```

Рис 3.5 – Вказання розташування вхідного файлу

Бажано перенести файл в папку з робочої області програми для швидшого пошуку. Також, якщо файл знаходиться в робочій області можна не вказувати повний шлях до розташування файлу, а вказати лише його назву та розширення.

Далі слід задати ім’я та розташування вихідному файлу (Рис 3.2).



```
package module;

import java.io.BufferedWriter;

public class FileOutput {
    public void WriteOutputToFile(String log) throws IOException {
        FileWriter fwriter = new FileWriter("output.txt", true);
        BufferedWriter bwriter = new BufferedWriter(fwriter);
        try (PrintWriter out = new PrintWriter(bwriter)) {
            out.println(log);
        }
    }
}
```

Рис 3.6 – Ім’я та розташування для створення вихідного файлу

Тепер ми можемо ввести параметри необхідні для обчислення, такі як висота та ширина кадру у відношенні до кроп-фактору, та фокусна відстань. Вони задаються в класі *ConstValues* (Рис. 3.3).

```java
package model;

public class ConstValues {
    public static final double frame_h = 0.22 / 1.62;
    public static final double frame_w = 0.14 / 1.62;
    public static final double F = 0.18;

}

```

Рис 3.7 – Задання константних даних. Параметрів фотокамери

І перед компіляцією слід вибрати режим в якому буде працювати программа – одна фотография, чи всі фотографії за логу фотозйомки (Рис 3.4).

```java
package main;

import java.io.IOException;

import engine.Controller;

public class Main {

    public static void main(String[] args) throws IOException {
        Controller controller = new Controller();
        // controller.OnePhoto(1);
        controller.MultiplePhotos();
    }
}
```

Рис 3.8 – Вибір режиму роботи

Обравши один режим, інший необхідно задокументувати. При виборі режиму однієї фотографії в параметри методу слід вказати порядковий номер фотографії в таблиці вхідного файлу.

Після чого можна компілювати код. Після компіляції, за вказаним шляхом буде створений вихідний файл с даними розрахунків.

ВИСНОВКИ

Перш за все слід відзначити тенденцію розвитку систем для аерозйомки з БПЛА. З розвитком технологій знімальних апаратів та виходом безпілотних дронів на масові ринки, тема зйомки поверхні землі с повітря стала значно популярнішею. Також сучасне програмне забезпечення здатне компенсувати багато проблем та неточностей в аерофотозйомці. Але слід відзначити декілька проблем:

- Спрощення конструкцій БПЛА та дронів, які виходять на масовий ринок, що викликає за собою меншу стабільність та обмеженість в вагопідйомності та стабільності літального апарату.
- Ціна все ще є значною, адже беспілотні літаки з модернізацією та вдосконаленням систем не доступні для масового ринку, а дрони не здатні задовольнити галузеві стандарти якості та точності.

Також слід зазначити, що представлена в роботі система не є ідеальною, проте вона здатна мінімізувати похибки в навігації та значно півдвищити ефективність та оптимізувати процес аерознімальних робіт підтримуючи якісні показники.