УДК 629.7.014-519:004(045)

Нічіков Є.П.
Чирков А.В.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВИЗНАЧЕННЯ ОБЛАСТІ БАЧЕННЯ КАМЕРИ ЦІЛЬОВОГО ПРИЗНАЧЕННЯ БЕЗПІЛОТНОГО ПОВІТРЯНОГО СУДНА

Національний авіаційний університет

jk@47.kiev.ua
artem.chyrkov@i.ua

Розроблено інформаційну технологію визначення області бачення камери цільового призначення безпілотного повітряного судна. Розроблено архітектуру програмного забезпечення, що дозволить отримувати вхідні дані для реалізації запропонованих методів як із локальних джерел (наприклад, локальні файли відео та логів телеметрії), так і з відповідних камер та сенсорів безпілотного повітряного судна або з симулятора

Ключові слова: обробка цифрового відео; безпілотне повітряне судно

Вступ

Актуальною на сьогоднішній день галуззю технології є безпілотні повітряні судна (БПС). Окрім суто технічних питань пов'язаних з конструкцією та властивостями літального апарата, важливою є задача обробки відеоданих з камер цільового призначення, зокрема, оперативне опрацювання відео в режимі реального часу. Серед питань, які можуть виникати при оперативній обробці відеоряду, у контексті БПС можна виділити визначення області бачення камери цільового призначення БПС, що може знайти впровадження у різних галузях цивільного та військового призначення.

Серед основних складностей - тестування та впровадження, зокрема, програмних рішень: тестування на реальних БПС, по-перше, вимагає значних фінансових коштів, по-друге, значних витрат часу та, по-третє, є ризикованим навіть в умовах полігону. На основі FlightGeat Flight Simulator [1] у НВЦБА «Віраж» НАУ було розроблено програмно-апаратний комплекс для тестування обладнання та програмного забезпечення [2]. Тестування на симуляторі не є ризикованим, але також вимагає деяких коштів та значних витрат часу.

Найбільш оптимальним варіантом за всіма вищенаведеними критеріями є локальне тестування на відео, збереженому з камери цільового призначення літального апарату або знайденому в мережі інтернет, та логах телеметрії, отриманих аналогічним чином. Переваги локального тестування очевидні. До найбільш суттєвих недоліків можна віднести той факт, що джерело вхідної інформації при такому способі не співпадає із джерелом при використанні готової системи на реальному БПС. При цьому будь-який метод обробки працює з абстрактними поняттями «кадр відео» та із конкретними значеннями параметрів телеметрії, незалежно від джерел їх отримання.

Аналіз публікацій та постановка задачі

При обробці відеоданих виникає потреба вибору технології, яка б дозволила проводити обробку потокового відео у режимі реального часу, була б досить гнучкою для реалізації розроблених алгоритмів та не потребувала спеціального обладнання.

Популярною при розробці програмного забезпечення для БПС [3, 4] є бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення із відкритим кодом – OpenCV, яка реалізована на C/C++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов [5].

Суттєво збільшити обчислювальну продуктивність завдяки використанню

графічних процесорів фірми NVIDIA [6] дозволяє програмно-апаратна архітектура паралельних обчислень CUDA [7]. Недоліком цієї технології є прив'язка до конкретного типу графічних процесорів. Для написання комп'ютерних програм, що пов'язані з паралельними обчисленнями на різних графічних та центральних процесорах, а також FPGA можна використовувати фреймворк OpenCL [8]. До складу фреймворку OpenCL входить мова програмування, яка базується на стандарті C99 [9], та інтерфейс програмування доданків (англ. API). OpenCL забезпечує розпаралелення на рівні інструкцій та на рівні даних та є реалізацією техніки GPGPU.

Поставимо за **мету даної роботи** розробити інформаційну технологію визначення області бачення камери цільового призначення безпілотного повітряного судна. При реалізації запропонованих методів необхідно використовувати таку архітектуру, що дозволить отримувати необхідні дані як із локальних джерел (наприклад, локальні файли відео та логів телеметрії), так і з відповідних камер та сенсорів БПС або з симулятора. При цьому кожна з реалізацій повинна подавати потоки даних на вхід методу обробки однаковим чином.

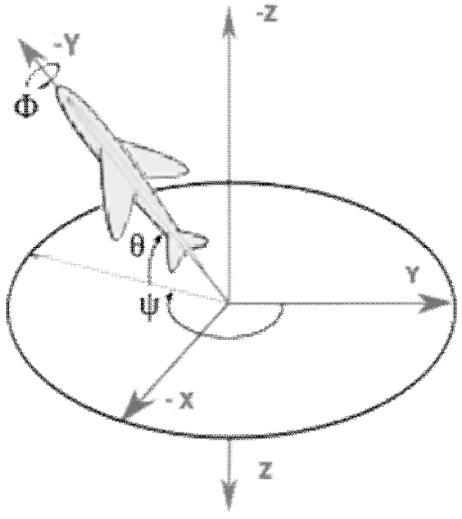

Рис. 1. Кути повороту літального апарату

Викладення основного матеріалу

Введемо тривимірну правосторонню систему координат, яка визначає положення БПС:

1) початок системи координат знаходиться у центрі мас літального апарату;
2) вісь Z спрямована до поверхні землі;
3) вісь Y спрямована на південь;
4) вісь X спрямована на схід.

У такій системі координат кутове положення літального апарату однозначно визначається 3 кутами: рискання (y , yaw), тангажу (q , pitch) та крену (g , roll). Додатні напрямки кутів зображені на рисунку (рис. 1). При нульових кутах ніс літака спрямований на північ.

Нехай (x, y, z) - координати точки P центра області бачення в заданій системі координат при нульових кутах повороту та висоті $z = 1$ (м). Знайдемо координати точки P' після повороту на довільні задані кути рискання, тангажу та крену. Для цього перейдемо до однорідних координат, тоді

$$P = (\frac{x}{h}, \frac{y}{h}, \frac{z}{h}, h),$$

де $h = 1$.

Обертання тривимірної точки навколо довільної вісі задається матрицями:

$$R_X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(q) & -\sin(q) & 0 \\ 0 & \sin(q) & \cos(q) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

$$R_Y = \begin{bmatrix} \cos(g) & 0 & -\sin(g) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(g) & 0 & \cos(g) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad (1)$$

$$R_Z = \begin{bmatrix} \cos(y) & -\sin(y) & 0 & 0 \\ \sin(y) & \cos(y) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Оскільки операція множення матриці на вектор не є комутативною, порядок повертання буде впливати на кінцевий результат. Враховуючи порядок множення матриць повороту кутів літального апарату отримаємо наступну формулу знаходження точки P':

1.

$$(x', y', z', 1) = (x, y, z, 1) * R_Z * R_X * R_Y . \qquad (2)$$

Відстань до центру кадру визначається з проекції точки P' на поверхню землі. Якщо задана висота l' (м), зсув уздовж осей X та Y буде визначатися наступними співвідношеннями:

2.

$$x'' = \frac{x'}{z'} * l' \quad , \quad y'' = \frac{y'}{z'} * l' \quad . \qquad (3)$$

Для знаходження координат центру кадру необхідно обрахувати зсув для поточних координат БПС. Наприклад, в проекції Меркатора [10]:

3.

$$(X'_M, Y'_M) = (X_M + \frac{x''}{S}, Y_M + \frac{y''}{S}),$$

$$S = C * \frac{cos(lat)}{2^{(zoom\_level+8)}}, \; C = 6378137 , \qquad (4)$$

де (X_M, Y_M) - поточні координати БПС в проекції Меркатора, lat - поточна широта БПС, $zoom\_level$ - рівень масштабу карти, C - радіус Землі на екваторі.

Визначення координат граничних точок кадру

Для розв'язання задачі визначення координат кутових точок кадру необхідно мати уявлення про їх координати при початкових умовах. Постає задача калібрування камери.

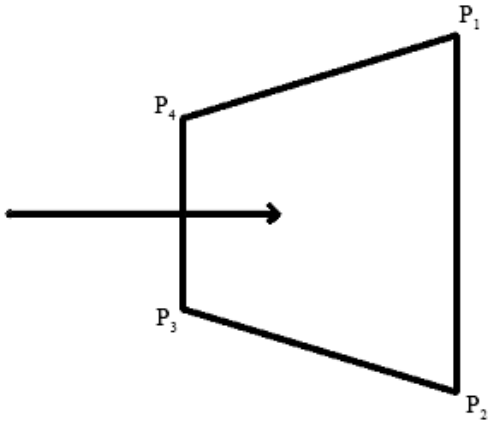

Рис. 2. Початкове положення камери БПС

Одним з способів є визначення координат (в метрах) при зйомці на стенді. На рисунку (рис. 2) наведено приклад отриманих даних. Якщо при стендовій зйомці висота відмінна від 1 м, необхідно провести нормалізацію координат

$$x_н = \frac{x}{z}, \; y_н = \frac{y}{z}.$$

Підставляючи (1) в формулу (2) отримаємо просторові координати граничних точок після повороту літака. Використовуючи (3) та (4) знайдемо координати області бачення камери.

У випадку, коли відомий кут f огляду камери та співвідношення сторін знімку F, граничні точки визначаються наступним чином:

$$P_1: \left(a * \cos(a), a * \sin(a)\right); \; P_2: \left(a * -\cos(a), a * \sin(a)\right);$$

$$P_1: \left(a * -\cos(a), a * -\sin(a)\right); \; P_1: \left(a * \cos(a), a * -\sin(a)\right);$$

де $a = tg\frac{f}{2}$, $a = \arcsin\left(\frac{1}{\sqrt{2a(F+1)}}\right)$.

Як було зазначено вище, будь-який метод обробки має працювати з абстрактними поняттями «кадр відео» та із конкретними значеннями параметрів телеметрії, незалежно від джерел їх отримання. В термінах об'єктно-орієнтованого програмування (ООП) наведені вимоги реалізуються виділенням абстрактного класу (якщо можливо, інтерфейсу) з відповід-

ним набором чисто віртуальних методів (pure virtual methods) та набором похідних класів із конкретними реалізаціями отримання вхідної інформації. Також за наявності необхідності незалежності потоків інформації різного типу (наприклад, відео та телеметрія) один від одного має зміст використання патерна програмування «Посередник» (англ. "Mediator pattern"). Даний патерн для даної задачі може бути використаний безпосередньо.

Для розробки програмної частини комплексу авторами даної статті використовуються мова програмування C++. Тестування програмного забезпечення проводилося на даних з борту БПС виробництва НВЦБА «Віраж» НАУ. У якості вхідних даних були використані координати літального апарату (довгота, широта, висота), кути нахилу літака (крін, тангаж, рискання) та координати кутових точок кадру при стендовій зйомці (рис. 2). Приклад визначення області бачення в деякий момент польоту представлено на рисунку (рис. 3).

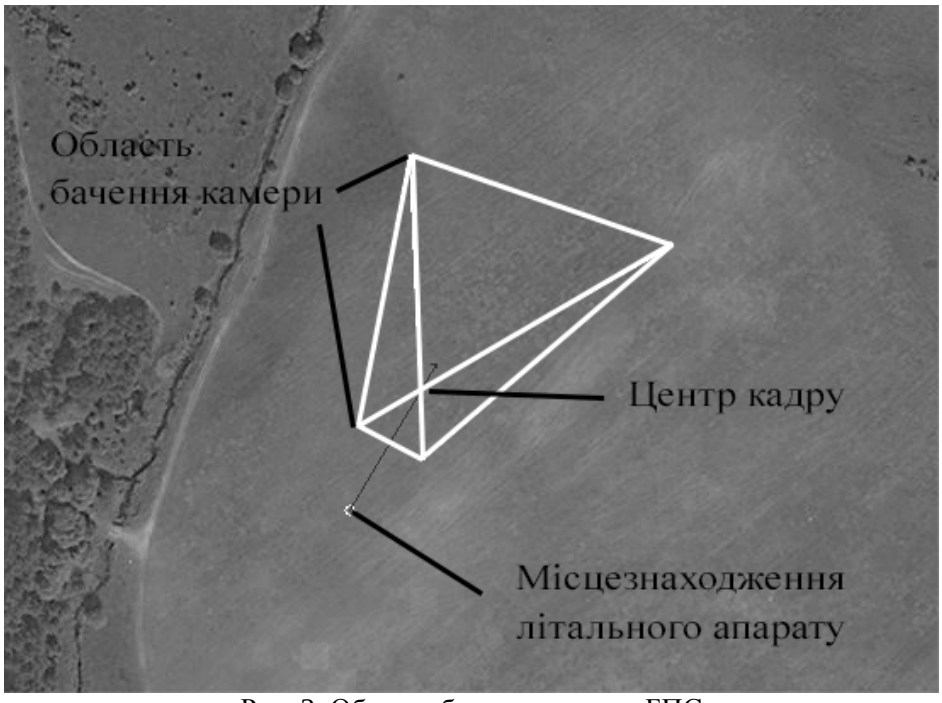

Рис. 3. Область бачення камери БПС

UML-діаграму класів для отримання та обробки вхідних даних наведено на рисунках (рис. 4, рис. 5).

Клас Application**,** точніше кажучи, похідні класи, що реалізують даний абстрактний клас (рис. 4), використовується для обробки кадрів відеопотоку:

- void init(int argc, const char* argv[]) – початкова ініціалізація, зокрема, обробка переданих параметрів командної строки;
- int run() – запустити реалізацію метода на відеопотоці. Повернути 0 (нуль) у випадку штатного завершення, або «не нуль» у випадку помилки;
- void frameCallback(cv::Mat&) та інші відповідні – обробити поточний кадр (передається як вхідний параметр метода).

Примітки.

1) Даний клас містить захищені поля для деяких сервісних цілей та реалізацію деяких методів за замовчанням, тому не є інтерфейсом у термінах ООП.
2) Віртуальні методи для обробки натискань клавіш реалізовані аналогічним чином та не наведені на UML-діаграмі з метою її спрощення.

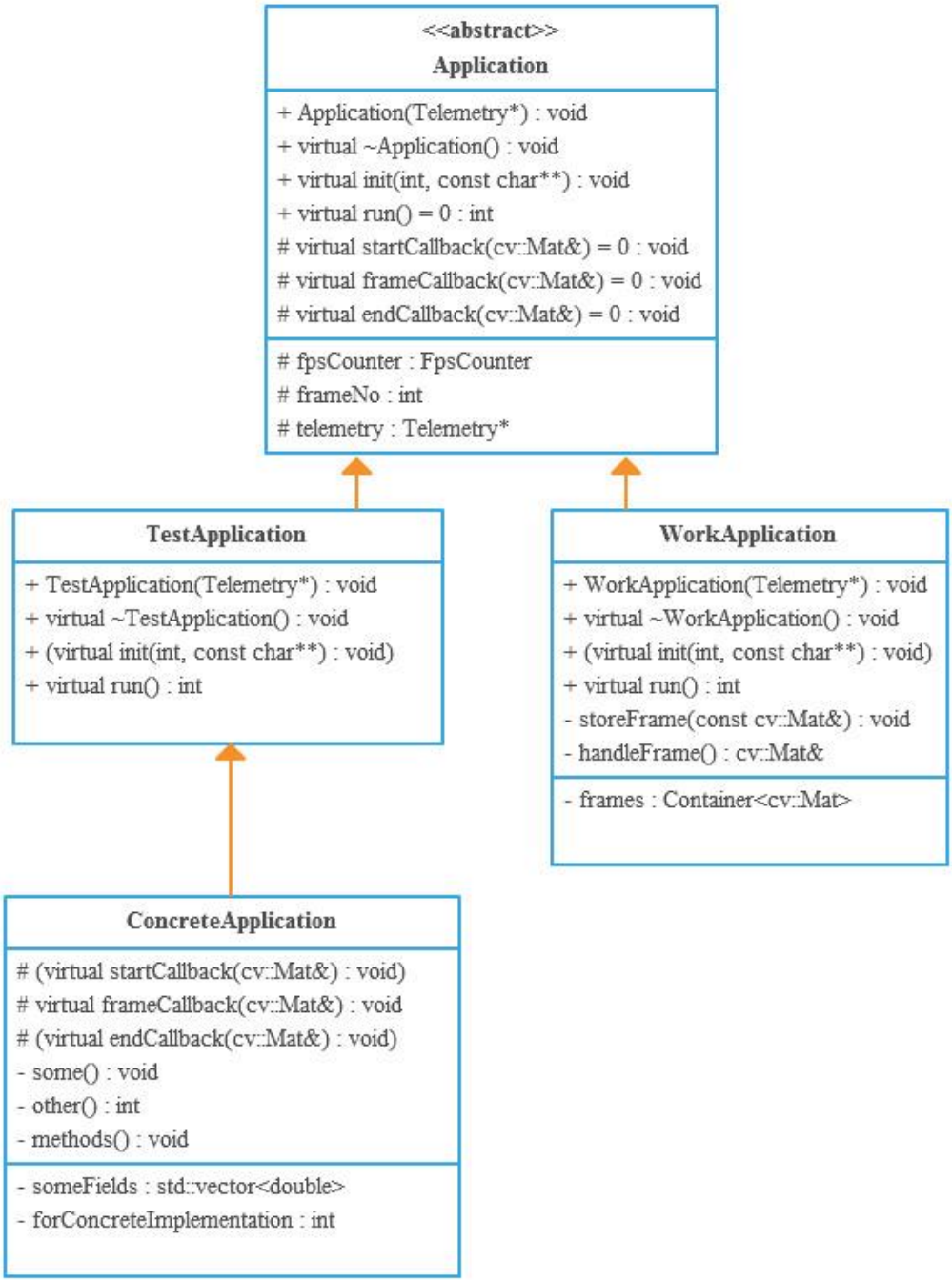

Рис. 4 UML-діаграма класів обробки відео потоку

На наступному рівні ієрархії класів знаходяться класи TestApplication, що реалізує роботу з локальними джерелами вхідних даних, та WorkApplication, що реалізую роботу з реальними джерелами.

На даному рівні виконується конкретна реалізація метода int run(), тобто фактично алгоритм отримання кадрів відеопотоку. Обов'язковою умовою реалізації є виклик віртуального метода void frameCallback(…) для кожного отриманого кадру.

Серед виявлених особливостей можна відмітити той факт, що завантаження кадрів із локального відеофайлу відбувається практично миттєво (затримка настільки мала, що нею можна знехтувати), а завантаження з реальної камери реального БПС (або симулятора) – із затримкою, якою не можна нехтувати відносно часу обробки кадру. Тому в реалізації

int WorkApplication::run() виникла необхідність зберігати отримані кадри в деякому контейнері–реалізації черги та брати їх для обробки з неї.

Третій рівень – конкретна реалізація конкретного метода обробки відеопотоку (клас ConcreteApplication). На даному рівні виконується конкретна реалізація метода void frameCallback(…) та аналогічних йому. При цьому за необхідності локального тестування даних клас реалізується як похідний від TestApplication, а при остаточній зборці програмного продукта (для роботи на реальному БПС або симуляторі) – від WorkApplication.

Із наведеного опису видно, що розбиття на три рівні ієрархії є очевидним та логічним (що є перевагою); при цьому перший рівень виконує роль інтерфейса, других рівень реалізує глобальну його частину, а третій рівень – локальну.

Клас Telemetry. (рис. 5) Як раніше було відмічено, вхідними даними є не тільки кадри відеопотоку, а і телеметрія. На БПС та симуляторі її параметри отримуються за деяким протоколом (напр., MAVLink [11]), а при локальному тестуванні завантажуються з файла логів.

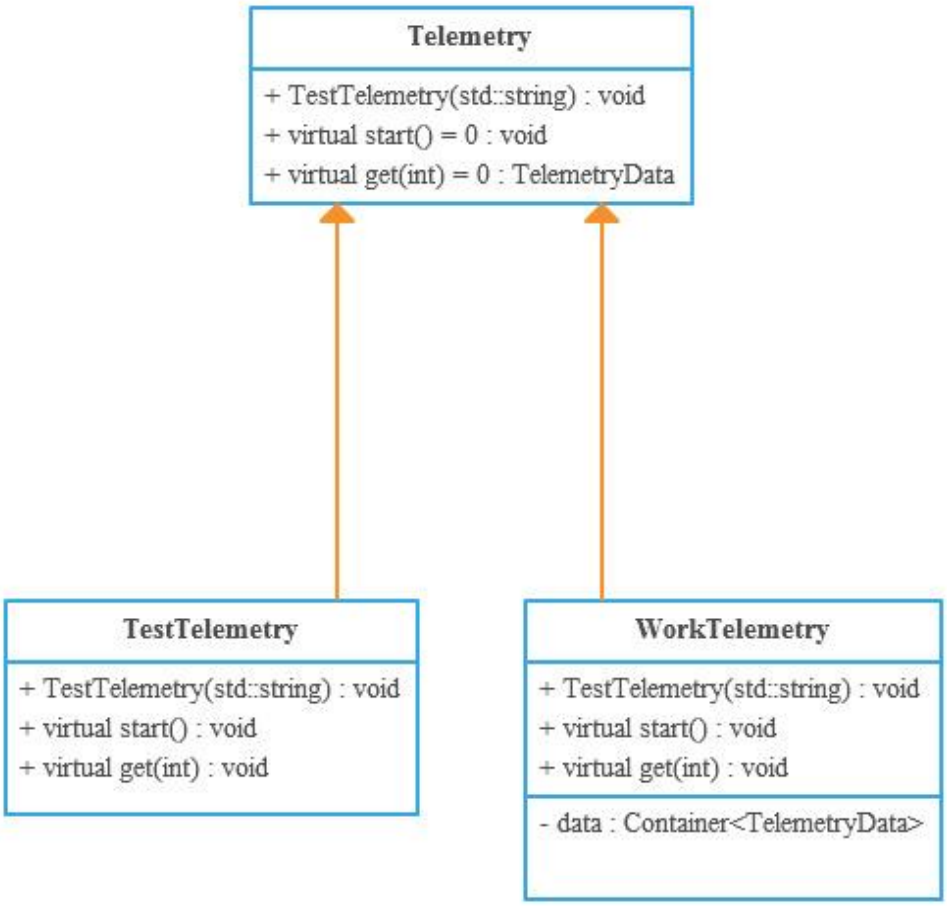

Рис. 5 UML-діаграма класів обробки телеметрії

Система класів для отримання телеметрії аналогічна першим двом рівням системи класів для отримання відеокадрів.

Деякі задачі потребують обидва вказані потоки одночасно. Отже, виникає задача їх одночасної і, головне, синхронізованої (телеметрія повинна відповідати кадру) передачі методу обробки.

Традиційним способом реалізації є використання патерна програмування «Посередник»: додається третій клас, через який перші два використовують один одного. Проте у даному випадку має зміст не використовувати вказаний підхід, а ре-

алізувати взаємодію класів безпосередньо.

Для цього умовно будемо вважати клас (ієрархію класів) Application «головною», а клас (ієрархію класів) Telemetry – допоміжною. При такому припущенні можна просто передати в реалізацію Application вказівник на реалізацію Telemetry (див. UML-діаграму) і викликати методи останньої з першої.

Висновки

Розроблено інформаційну технологію визначення області бачення камери цільового призначення безпілотного повітряного судна.

Розроблено архітектуру програмного забезпечення, що дозволить отримувати необхідні дані як із локальних джерел (наприклад, локальні файли відео та логів телеметрії), так і з відповідних камер та сенсорів БПС або з симулятора. При цьому кожна з реалізацій повинна подавати потоки даних на вхід методу обробки однаковим чином.

Список літератури

1. FlightGear Flight Simulator [Електронний ресурс] : http://www.flightgear.org/

2. О.В. Коваль. Система моделювання польоту безпілотного повітряного судна та тестування його бортового обладнання / О.В. Коваль, Є.П. Нічіков, О.С. Васильєв ISSN 2075-0781. Наукоємні технології. № 1 – К.: НАУ, 2013 – С. 78-81

3. Б.А. Алпатов, П.В. Бабаян, Ю.С. Коблов, В.С. Муравьев, В.В. Стротов, А.Б. Фельдман. Программный комплекс для автоматизации научных исследований в области создания перспективных систем навигации беспилотных летательных аппаратов по данным видеонаблюдения / ISSN 1995-4565. Вестник РГРТУ. № 2 (выпуск 40). Рязань, 2012

4. Hing, J.T., Sevcik, K.W., Oh, P.Y. Improving Unmanned Aerial Vehicle Pilot Training and Operation for Flying in Cluttered Environments / Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on 10-15 Oct. 2009. – pp. 5641 - 5646

5. Bradsky G., Kaehler A. Learning OpenCV — O'Reilly, 2008. — С. 1 — ISBN 978-0-596-51613-0

6. NVIDIA [Електронний ресурс] : http://nvidia.com/

7. CUDA ZONE [Електронний ресурс] : https://developer.nvidia.com/category/zone/cuda-zone

8. OpenCL [Електронний ресурс] : http://opencl.ru/

9. ISO/IEC 9899:1999/Cor 3:2007 от 2007-11-15 [Електронний ресурс] : http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=50510

10. ArcGIS Help 10.1 - Проекция Меркатора [Електронний ресурс] http://resources.arcgis.com/ru/help/main/10.1/index.html#//003r00000038000000

11. MAVLink Micro Air Vehicle Communication Protocol [Електронний ресурс] : http://qgroundcontrol.org/mavlink/start

Статтю подано до редакції 02.12.2015