Тумуров Э.Г., Шелехов В.И.

Постановка задачи для гибридной реактивной системы на примере задачи управления движением квадрокоптера по заданной траектории

Квадрокоптер AR.Drone – беспилотный летательный аппарат, управляемый дистанционно. На примере задачи управления движения квадрокоптера по заданной траектории исследуются общие формы постановки задачи для гибридной реактивной системы.

1. Введение

Существуют следующие классы программ: программы-функции, программы-процессы, трансляторы, операционные системы и другие [1]. Первые два класса покрывают более 90% всех программ. Технология построения надежных и эффективных программ-функций разработана в рамках исследований по предикатному программированию [3-8]. Для класса программ-процессов предложена технология автоматного программирования [1], отличная от switch-технологии А.А. Шалыто [15, 11]. В работе [1] представлен базис технологии автоматного программирования: набор языковых конструкций и основные методы построения программ. Дальнейшее развитие технологии требует учета специфики разных подклассов в классе программ-процессов.

Гибридные системы определяют подкласс реактивных систем, которые составляют наибольший подкласс в классе программ-процессов. Гибридные системы обычно рассматриваются в контексте дедуктивной верификации, где наработано большое число разных моделей программ. Их можно лишь частично использовать при разработке модели гибридных систем, ориентированной на технологию программирования.

В настоящей работе исследуется начальная часть разработки гибридной системы от постановки задачи до выбора решения. На этой стадии обычно фиксируются исходные данные задачи и требуемые результаты, определяются набор требований к создаваемой программе. Эту информацию принято называть спецификацией программы. Одна из целей данной работы – определить общую форму спецификации гибридных систем как специализацию общей формы спецификации программ-процессов.

Исследование формы постановки задачи для гибридной системы проводится на примере задачи управления движением беспилотным летательным аппаратом – квадрокоптером AR.Drone.

2. Классы программ

Обычно рассматриваются классификации по назначению программ. Здесь же определяется классификация программ по их внутренней организации. Цель классификации – разработка адекватной технологии программирования для каждого класса программ. Теория программ каждого класса должна определять методы спецификации, верификации (в широком смысле), моделирования и эффективной реализации программ. Базисом классификации являются:

- внешняя форма программы (интерфейс с окружением);
- минимальное ядро (набор конструкций) языка программирования;
- формы определения спецификации программы;
- виды условий корректности программы относительно спецификации.

Генеральная классификация определяет два класса программ:

- невзаимодействующие программы (или программы-функции);
- программы-процессы.

Данные два класса составляют более 90% всех программ. Имеются другие, более сложные классы, например, языковые процессоры и операционные системы; они находятся на метауровне по отношению к первым двум классам. Языковые процессоры – это интерпретаторы программ, компиляторы, оптимизаторы, трансформаторы и т.д. Приведенная классификация не покрывает всего спектра программ и различных особенностей, например такой, как тесная интеграция программы с данными.

Класс *невзаимодействующих программ* или класс *программ-функций*. Программа принадлежит этому классу, если она не взаимодействует с внешним окружением. Точнее, если возможно перестроить программу таким образом, чтобы все операторы ввода данных находились в начале программы, а весь вывод собран в конце программы, то такая программа относится к классу невзаимодействующих программ. Программа обязана всегда завершаться, поскольку бесконечно работающая и невзаимодействующая программа бесполезна. Следовательно, программа определяет функцию, вычисляющую по набору входных данных (аргументов) некоторый набор результатов. Класс программ-функций, по меньшей мере, содержит программы для задач дискретной и вычислительной математики.

Программа-функция реализует решение (алгоритм) некоторой математической *задачи*. Решение задачи строится на базе *логики решения* – набора математических свойств (теорем). Программирование – это реализация логики решения в конструкциях языка программирования. Понимание программы реализуется в процессе сопоставления кода программы с логикой решения. В качестве иллюстрации рассмотрим логику решения для простейшей программы умножения натуральных чисел a и b через операцию сложения:

$$a * b = b + (a - 1) * b, \quad 0 * b = 0 . \tag{1}$$

Ниже приведены логическая, функциональная и императивная программы, являющиеся реализациями логики решения (1).

$$0 * b \to 0; \;\; a * b \to b + (a - 1) * b \tag{2}$$

```
nat mult(nat a, b) { if (a = 0) 0 else b + mult(a – 1, b) }        (3)
```

```
nat c = 0; while (a != 0) { c = c + b; a = a – 1 }                 (4)
```

Различия логической программы (2) и логики (1) чисто синтаксические. Фактически они тождественны. Функциональная программа (2) и логика (1) отличаются. Однако они также тождественны – программа (2) легко транслируется в логику (1). Императивная программа (4) также построена из логики (1), однако понять ее и установить тождественность с логикой (1), – нетривиальная задача.

Спецификация программы S(x: y) с аргументами x и результатами y определяется предусловием P(x) и постусловием Q(x, y). Тотальная корректность программы относительно спецификации определяется формулой:

$$P(x) \Rightarrow \forall y.\ [\ L(S(x{:}\ y)) \Rightarrow Q(x, y)\ ]\ \&\ \exists y.\ L(S(x{:}\ y))$$

где L(S(x: y)) – *логика программы* S(x: y) – сильнейший предикат, истинный при завершении исполнения программы.

Класс *программ-процессов* (*автоматных программ*). Программа данного класса является автоматной программой, определяемой в виде конечного автомата и состоящей из набора сегментов кода. Вершина автомата соответствует некоторому *управляющему состоянию*. Ориентированная гипердуга автомата соответствует некоторому сегменту кода и связывает одну вершину с одной или несколькими другими вершинами. Исполнение сегмента завершается переходом на начало другого сегмента.

Взаимодействие с внешним окружением автоматной программы реализуется через прием и посылку *сообщений*, а также через *разделяемые переменные*, доступные для чтения и записи в данной программе, а также в других программах из окружения программы. *Состояние* автоматной программы определяется значениями набора переменных, модифицируемых в программе, за исключением локальных переменных. Управляющее состояние программы идентифицирует текущий исполняемый сегмент.

Важнейшим подклассом автоматных программ являются контроллеры систем управления в аэрокосмической отрасли, энергетике, медицине, массовом транспорте и др. отраслях. На

каждом шаге вычислительного цикла контроллер получает входную информацию из окружения и обрабатывает ее. Результаты вычисления используются для передачи управляющего сигнала в окружение контроллера.

Спецификация автоматной программы включает:

- инвариант в каждом управляющем состоянии, определяемый на переменных состояния;
- свойства безопасности (safety), живости (liviness), справедливости (fairness) и др. в виде формул на языке темпоральной логики.

Свойство устойчивости системы управления – подвид свойства безопасности.

3. Гибридная управляющая система

Реактивные системы – программы-процессы, реагирующие на внешние события. Гибридная система – реактивная система, соединяющая дискретное и непрерывное поведение. Часть переменных состояния гибридной системы соответствует непрерывным параметрам, изменение которых реализуется независимо от программы гибридной системы по определенным законам, обычно формулируемым в виде дифференциальных уравнений.

В нашем случае гибридная система является также системой управления. Программа системы управления имеет следующую структуру:

```
process ControlSystem(... : ... #exit) {
    M:      inv <инвариант>;
            receive currentData(in) {
               Step(in, s: s', out #M: #exit);
               send controlData(out);
            }
}
```

На очередном шаге работы программа получает данные in – текущие значения параметров объекта управления. В соответствии с целью управления программа-контроллер Step вычисляет управляющие данные out для передачи команд с целью корректировки поведения объекта управления. Вычисление требуемых значений par параметров, соответствующих цели управления, обычно реализуется решением дифференциальных уравнений E(in, par, out), определяющих закон функционирования управляемого объекта. Контроллер также модифицирует состояние s, отражающее информацию о предыдущих шагах работы системы управления; в простейшем случае s′ = in.

Более типичным является запуск контроллера через определенный промежуток времени. В этом случае сообщение currentData(in) поставляет последние значения параметров, регулярно поступающих от объекта управления.

Необязательный выход #exit может использоваться, например, в случае достижения цели управления.

4. Квадрокоптер AR.Drone

Рассматривается задача управления движением беспилотным летательным аппаратом – квадрокоптером AR.Drone по заданной траектории с постоянной скоростью на постоянной высоте от точки A до точки B. Траектория задается уравнением $l(x, y) = 0$, высота z является постоянной.

Параметры квадрокоптера. Положение в пространстве характеризуется координатами x, y, z центра масс аппарата в неподвижной декартовой системе координат, а ориентация – тремя углами Эйлера относительно системы координат (x_b, y_b, z_b), жестко связанной с аппаратом. Ось x_b направлена вдоль продольной оси, y_b – вдоль поперечной, z_b – вдоль вертикальной. Начало координат совпадает с центром масс аппарата. Общепринятые обозначения:

- ψ – угол рыскания – угол поворота вокруг оси z_b; (-беск; +беск);

- φ – угол крена – угол поворота вокруг оси x_b; (-pi/2; +pi/2);
- θ – угол тангажа – угол поворота вокруг оси y_b; (-pi/2; +pi/2).

Другие параметры: скорости v_{xb}, v_{yb} в проекции на оси x_b и y_b и угловые скорости $\varphi^$, $\theta^$ в системе координат (x_b, y_b, z_b). Таким образом, полный набор параметров in = x, y, z, ψ, φ, θ, v_{xb}, v_{yb}, $\varphi^$, $\theta^$, причем углы ψ, φ, θ определяются в абсолютной системе координат. Аппарат 200 раз в секунду передает параметры на устройство дистанционного управления, кроме координат x, y, z, которые поступают от внешних камер.

Управляющие параметры квадрокоптера. Аппарат может принимать команды управления каждые 30мс. Команда управления определяется последовательностью управляющих параметров out = φ, θ, $\psi^$, $z^$, где $z^$ – вертикальная скорость. Значения этих параметров задаются в ограниченных диапазонах. Бортовая система интерпретирует команды управления и автоматически реализует стабилизацию полета.

5. Обзор работ

Беспилотные квадрокоптеры представляют большой практический интерес для практических целей фото/видео наблюдения, доставки грузов благодаря отличной устойчивости, компактности, гибкости управления, достаточно тихой работы.

В последние 5 лет появились доступные коммерческие аппараты от производителей Parrot, 3D Robotics, Walkera, Gryphon Dynamics, Freefly systems, Aeryon и др. Развиваются проекты с открытым исходным кодом и открытым “железом” Arducopter, AeroQuad, OpenPilot, Crazyflie, и др. Crazyflie создали помещающийся на ладони квадрокоптер.

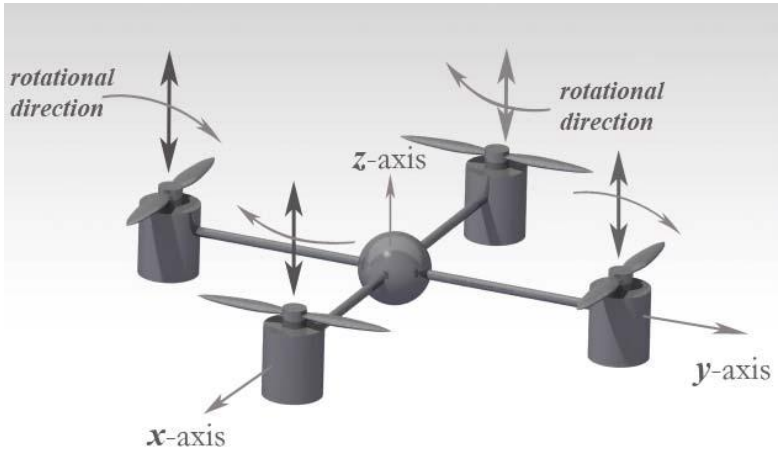
Ведущие лаборатории по изучению динамики и систем управления квадрокоптеров Aerospace Controls Lab of MIT, Flying machine arena of ETH Zurich, GRASP Lab of University of Pennsylvania занимаются исследованием оптимального управления максимально быстрых маневров квадрокоптеров, управлением группы квадрокоптеров, квадрокоптеров с изменямыми углами атаки лопастей.

В работе [24] исследуется применимость линейных уравнений движения для квадрокоптера в задаче следования по траектории (численный эксперимент). В идеальных условиях квадрокоптер достаточно хорошо следовал траектории, но при добавлении ветра и ограничений на тягу роторов результаты оказались неудовлетворительными.

Способы управления квадрокоптером для парящих режимов полета (near-hover operation) хорошо изучены [21, 25, 27]. В большинстве работ считается, что аппарат должен следовать непосредственно по заданной траектории. Аспект осуществимости полета по траектории начал изучаться позже. Работы [18, 19, 20] предлагают алгоритмы, разделяющие задачу планирования на две части. Сначала строятся траектории без временных данных из примитивов движения (линии, сплайны). После этого траектория параметризуется по времени так, чтобы скорость движения удовлетворяла критериям осуществимости. В [21] представлена методика с использованием нелинейного программирования и генетических алгоритмов для минимизации времени полета. В [17] решается задача о перемещении в некоторую точку из любого положения и состояния максимально быстрым способом, алгоритм достаточно быстр для управления и коррекции траектории в реальном времени (50Гц).

Предложен эффективный метод построения быстрой траектории полета квадрокоптера в сложной конфигурации препятствий [23, 26].

Управление квадрокоптером AR.DRONE исследовалось в Новосибирске [28, 29]. Построена модель бортовой системы управления, система управления и измерения положения в пространстве, метод фильтрации шумов с помощью фильтра Калмана.



6. Анализ постановки задачи управления движением квадрокоптером

Постановка задачи для программы-функции определяет математическую задачу и эксплицируется спецификацией в виде предусловия и постусловия. Решение задачи определяет логику решения, которая записывается программистом в конструкциях языка программирования. Данная схема не годится для программ-процессов.

Исходными данными задачи управления движением квадрокоптером являются исходные и управляющие параметры, определенные в разделе 4. Данную задачу можно было бы представить как задачу оптимизации по достижении наилучшего приближения траектории полета к заданной траектории. Более слабой постановкой является задача достижения устойчивого движения, при котором в случае отклонений от заданной траектории реализуется гарантированное быстрое возвращение к заданной траектории.

По данной задаче пока не предлагается та или иная форма постановки задачи. Определить такую форму – задача дальнейшего исследования. Рассматриваемая задача получилась недоопределенной, поскольку поставлена в целях поиска адекватного математического решения. Иначе говоря, задача имеет модельный характер. Реальная практическая задача обычно включает более точные условия и ограничения. Например, возможно наличие препятствий при движении по траектории. Качественно иной и гораздо более сложной является задача управления полетом в лабиринте пещер. Ограничения могут быть также вызваны предельно допустимым отклонением от траектории, возникающие например, при съемке местности вдоль заданной траектории.

Работа выполнена при поддержке РФФИ, грант № 12-01-00686.

Список литературы

1. Шелехов В.И. Язык и технология автоматного программирования для разработки систем управления // Тр. 15-й межд. конф. «Проблемы управления и моделирования в сложных системах». — Самара, Самарский научный центр РАН, 2013. – С. 554-564.
2. Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998. http://is.ifmo.ru/books/switch/1
3. Карнаухов Н.С., Першин Д.Ю., Шелехов В.И. Язык предикатного программирования Р. – Новосибирск, 2010. – 42с. – (Препр. / ИСИ СО РАН; N 153).
4. Шелехов В.И. Разработка программы построения дерева суффиксов в технологии предикатного программирования. — Новосибирск, 2004. — 52с. — (Препр. / ИСИ СО РАН; N 115).
5. Shelekhov V. I. 2011. Verification and Synthesis of Addition Programs under the Rules of Correctness of Statements. *Automatic Control and Computer Sciences*. Vol. 45, No. 7, 421–427.
6. Шелехов В.И. Верификация и синтез эффективных программ стандартных функций в технологии предикатного программирования // Программная инженерия, 2011, № 2. — С. 14-21.
7. В.А. Вшивков, Т.В. Маркелова, В.И. Шелехов. Об алгоритмах сортировки в методе частиц в ячейках // Научный вестник НГТУ, Т.4(33), с. 79-94, 2008.

8. Шелехов В.И. Разработка и верификация алгоритмов пирамидальной сортировки в технологии предикатного программирования. – Новосибирск, 2012. – 30с. – (Препр. / ИСИ СО РАН. N 164).
9. Cooke D. E., Rushton J. N. Taking Parnas's Principles to the Next Level: Declarative Language Design // Computer, Vol. 42, no. 9. – 2009. – P.56-63.
10. Hewitt C., Bisshop P. and Steiger R. *A Universal Modular Actor Formalism for Artificial Intelligence* // 3rd IJCAI, Stanford, CA, 1973, pp. 235-245.
11. Автоматное программирование. http://ru.wikipedia.org/wiki/Автоматное_программирование
12. Lamport L. Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers / Addison-Wesley. – 1999.
13. Abrial J.-R. Modelling in Event-B: System and Software Engineering / Cambridge Univ. Press, 2010.
14. Specification and description language (SDL). *ITU-T Recommendation Z.100* (03/93).
15. Поликарпова Н.И., Шалыто А.А. Автоматное программирование / СПб.: Питер. 2009, 176с. http://is.ifmo.ru/books/_book.pdf
16. Шелехов В.И. Предикатное программирование. Учебное пособие. – НГУ, Новосибирск, 2009. – 109с.
17. M. Hehn, R. D'Andrea. Quadrocopter Trajectory Generation and Control. // IFAC World Congress, 2011, pp. 1485-1491.
18. G.M. Hoffman, S.L. Waslander, C.J. Tomlin. Quadrotor Helicopter Trajectory Tracking Control. // In Proc. AIAA Guidance, Navigation and Control Conf, 2008.
19. I.D. Cowling, O.A. Yakimenko, J.F. Whidborne. A Prototype of an Autonomous Controller for a Quadrotor UAV. // In European Control Conference, 2007.
20. Y. Bouktir, M. Haddad, T. Chettibi. Trajectory Planning for a Quadrotor Helicopter. // In Mediterranean Conference on Control and Automation. IEEE, June 2008.
21. L. Lai, C. Yang, C. Wu. Time-Optimal Control of a Hovering Quad-Rotor Helicopter. // Journal of Intelligent and Robotic Systems, 45(2), pp. 115-135, June 2006.
22. S. Shen, Y. Mulgaonkar, N. Michael, V. Kumar. Vision-Based State Estimation and Trajectory Control Towards High-Speed Flight with a Quadrotor. // Proceedings of Robotics: Science and Systems IX, June 2013.
23. R. Charles, A. Bry, N. Roy. Polynomial Trajectory Planning for Quadrotor Flight. // Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, 2013.
24. K. Miller. Path tracking control for quadrotor helicopters, 2008.
25. D. Xu, L. Wang, G. Li, L. Guo. Modeling and Trajectory Control of a Quad-rotor UAV // ICCASM-12, Advances in Intelligent Systems Research, July 2012.
26. D. Mellinger, V. Kumar. Minimum Snap Trajectory Generation and Control for Quadrotors. // Robotics and Automation (ICRA), 2011 IEEE International Conf.
27. R.W. Beard. Quadrotor Dynamics and Control. // Brigham Young University, Feb 2008
28. С.А. Белоконь, Ю.Н. Золотухин, А.С. Мальцев, А.А. Нестеров, М.Н. Филиппов, А.П. Ян. Управление параметрами полёта квадрокоптера при движении по заданной траектории. // Автометрия, 2012, С.32-42.
29. С.А. Белоконь, Ю.Н. Золотухин, К.Ю. Котов, А.С. Мальцев, А.А. Нестеров, М.Н. Филиппов, А.П. Ян. Управление квадрокоптером AR.DRONE при движении по заданной траектории. // Труды XV Международной конференции «Проблемы управления и моделирования в сложных системах», 25-28 июня 2013 г., Самара, С.506-514.