УДК 618.518.5

Требования к системе управления квадрокоптером

Тумуров Э.Г. (Институт систем информатики СО РАН),

Шелехов В.И. (Институт систем информатики СО РАН,

Новосибирский государственный университет)

Определяется типовая архитектура системы управления вида human-in-the-loop. Набор функциональных требований к системе управления беспилотным 4-х винтовым вертолетом - квадрокоптером – определяется как конкретизация типовой архитектуры. Рассматривается две задачи: управление полетом квадрокоптера по заданной траектории и слежение за движущимся объектом.

Ключевые слова: *Беспилотный летательный аппарат, квадрокоптер, human-in-the-loop, система управления, определение требований, автоматное программирование.*

1. Введение

Квадрокоптер – беспилотный 4-х винтовой вертолет. Для квадрокоптера предусмотрено два вида управления: ручное, осуществляемое человеком (оператором) дистанционно через Интернет, и автоматическое программное на базе интерфейса, определяющего систему команд управления квадрокоптером.

В настоящей работе определяется общая постановка задачи управления квадрокоптером, интегрирующая различные аспекты автоматического управления квадрокоптером [1-2, 9-10, 12, 14-15, 17-18, 20-21, 24-29, 31]. Вместе с традиционной задачей движения по заданному маршруту рассматривается новая задача слежения за движущимся объектом. Описывается типовая архитектура системы управления вида human-in-the-loop [19] с интеграцией автоматического и ручного управления. Спецификация задачи управления квадрокоптером определяется в виде набора требований.

Контроллер системы управления является реактивной системой. *Определение требований* – первый этап конструирования реактивной системы. *Требования* — совокупность утверждений относительно свойств разрабатываемой реактивной системы. Одним из видов требований являются *функциональные требования*, определяющие поведение программы. Их наиболее популярной формой являются *сценарии использования* (*use case*) [6]. В нашем подходе сценарии использования представлены в виде правил на языке продукций [8],

обычно применяемом для систем искусственного интеллекта. Это простой язык с высокой степенью декларативности. Спецификация автоматной программы в виде набора требований компактна и легко транслируется в автоматную программу, что позволяет использовать язык спецификации требований как язык автоматного программирования [4].

Типовая архитектура системы управления вида human-in-the-loop представлена в разделе 2 автоматными программами автоматического контроллера и мониторинга. Программы специфицированы набором требований. Язык требований кратко описывается в разд. 2.1; его полное описание доступно в [4]. Задача управления беспилотным летающим аппаратом (БПЛА) определена в общем виде в подразделе 3.1 как конкретизация типовой архитектуры. В свою очередь система управления квадрокоптером (подраздел 3.1.1) характеризуется конкретизацией входных и управляющих параметров системы управления БПЛА. Постановка задачи управления движением квадрокоптера по заданной траектории (подраздел 3.2) определяет задание маршрута движения, набор возможных ограничений и препятствий. В подразделе 3.3 описана постановка задачи слежения за движущимся объектом с определением параметров и разнообразных аспектов. В обзоре работ (раздел 4) описываются различные подходы в постановке задачи управления квадрокоптером.

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

2. Типовая архитектура системы управления

Контроллер системы управления является реактивной системой. Для спецификации и реализации реактивных систем разработана технология автоматного программирования [4]. *Автоматная программа* определяется в виде конечного автомата и состоит из нескольких *сегментов кода*. Вершина автомата – *управляющее состояние* программы. Ориентированная гипердуга автомата соответствует некоторому сегменту кода и связывает одну вершину с одной или несколькими другими вершинами. Исполнение сегмента завершается оператором перехода на начало другого сегмента. Взаимодействие автоматной программы с внешним окружением реализуется через прием и посылку *сообщений*. *Состояние* автоматной программы определяется значениями набора переменных, модифицируемых в программе.

Реактивная система реализует взаимодействие с внешним окружением программы и реагирует на определенный набор событий (сообщений) в окружении программы. Подклассом реактивных систем являются *гибридные системы*, соединяющие дискретное и непрерывное поведение. Часть переменных состояния гибридной системы соответствует непрерывным параметрам (время, координаты в пространстве и др.), изменение которых реализуется независимо от программы гибридной системы по определенным законам,

обычно формулируемым в виде дифференциальных уравнений. Важнейшим подклассом гибридных систем являются контроллеры систем управления.

Система управления реализует взаимодействие с *объектом управления* для поддержания его функционирования в соответствии с поставленной целью. Системы управления используются в аэрокосмической отрасли, энергетике, медицине, массовом транспорте и др. отраслях. На каждом шаге вычислительного цикла *контроллер системы управления* получает входную информацию из окружения и обрабатывает ее. Результаты вычисления используются для передачи управляющего сигнала, воздействующего на объект управления.

Большинство систем управления являются *системами реального времени*, определяющими набор мягких и жестких временных ограничений на взаимодействие с окружением. В системах с жестким ограничением непредоставление результатов вычислений к определенному сроку является фатальной ошибкой.

Наряду с управлением в автоматическом режиме возможно наличие *ручного управления*, осуществляемого человеком – *оператором*, иногда осуществляемого дистанционно через Интернет [29]. Чтобы ручное управление было эффективным, необходима реализация адекватной визуализации процесса управления. Автоматическое и ручное управление могут существовать раздельно без интеграции между собой. Определим типовую архитектуру системы управления вида human-in-the-loop [19], в которой автоматическое управление объектом интегрировано с ручным управлением. Контроллер системы управления состоит из двух параллельных процессов, определяемых автоматными программами AutoControl и Advisor. Архитектура системы управления представлена набором требований (разд. 2.2 и 2.3).

2.1. Язык требований

Требование определяет один из вариантов функционирования автоматной программы и имеет следующую структуру:

$$\langle\text{условие}_1\rangle, \langle\text{условие}_2\rangle, \ldots, \langle\text{условие}_n\rangle \rightarrow \langle\text{действие}_1\rangle, \ldots, \langle\text{действие}_m\rangle$$

Условиями являются: управляющие состояния, получаемые сообщения, логические выражения. *Действиями* являются: простые операторы, вызовы программ, посылаемые сообщения и итоговые управляющие состояния. Семантика требования следующая: если в данный момент времени истинны все условия в левой части требования, то последовательно исполняется набор действий в правой части.

Вызов программы записывается в виде A(x: y), где A – имя программы, x – список аргументов вызова, y – список результатов. Неблокированный прием сообщения реализуется

конструкцией m(y), где m – имя сообщения, y – список переменных. Значение конструкции – **true**, если из окружения получено сообщение с именем m; при этом переменным y присваиваются значения параметров сообщения m. Оператор перехода **goto** M записывается в виде #M.

Полное описание языка требований доступно в работе [4]. Описание требований контроллеров AutoControl и Advisor сопровождается содержательными пояснениями.

2.2. Автоматический контроллер

Контроллер AutoControl реализует циклический пошаговый процесс автоматического управления объектом. На каждом шаге контроллер определяет команду, которой нужно воздействовать на объект для достижения цели управления.

Состояние. Набор *переменных состояния* системы управления обозначается через s. В типичном случае состояние определяется: координатами объекта управления в пространстве, текущим временем и другими параметрами управления.

Окружение. Запуск контроллера, находящегося в спящем режиме, реализуется сообщением start. Срабатывание очередного шага работы контроллера происходит через сообщение next(in), где in – набор *входных параметров*, поступающих от объекта управления. Управляющее воздействие на объект управления реализуется посылкой сообщения control(com), где com – команда управления объектом с целью корректировки его поведения. Команда com определяет *вид команды* и набор параметров. Сообщение stop используется для перевода контроллера в спящий режим.

Управляющие состояния:

- idle – режим ожидания начала работы (спящий режим);
- run – основной рабочий режим.

Требования.

idle, start → run
run, next(in) → Step(in, s: s', com), control(com)
run, stop → idle

В управляющем состоянии idle контроллер ожидает сообщения start от процесса Advisor. При получении сообщения start контроллер переходит в основной режим – управляющее состояние run. В этом режиме ожидается сообщение next(in), запускающее программу Step. Она анализирует информацию in о текущем состоянии объекта управления и вычисляет управляющую команду com для модификации поведения объекта в соответствии с целями

управления. В случае перехода на режим ручного управления посылается сообщение stop, переводящее контроллер в спящий режим.

Приведенные требования транслируются в следующую автоматную программу:

```
Программа. process AutoControl(…) {
              idle: if (start) #run
                    else #idle
              run:  if (next(in)) { Step(in, s: s', com);
                                    send control(com);
                    } else if (stop) #idle
                    #run
            }
```

Сообщение next(in) обычно генерируется через определенный фиксированный промежуток времени с последними значениями параметров in, регулярно поставляемыми от объекта управления.

Задачей программы Step является достижение целевых значений параметров par объекта управления. По значениям in и par требуется вычислить соответствующие значения параметров команды com. Обычно это реализуется решением дифференциальных уравнений E(in, par, com), определяющих закон функционирования управляемого объекта. В дополнении к этому, возможны случайные воздействия на объект управления – *возмущения*; например, ветер. Информация о возникающих возмущениях может накапливаться в состоянии s в целях адекватной реакции.

Система управления должна быть *устойчивой*: при отсутствии возмущений ее параметры in должны приближаться к требуемым значениям par, а при ограниченных возмущениях параметры in должны оставался в ограниченной зоне вблизи этих значений [28]. Свойство устойчивости является подвидом свойства безопасности реактивной системы.

2.3. Мониторинг процесса управления

Автоматическое и ручное управление может проводиться одновременно лишь в очень редких случаях. Переход с автоматического управления на ручное предполагает отключение автоматического управления. Ручное управление блокируется при переходе на автоматическое управление, однако визуализация процесса управления для оператора сохраняется. Переключение с одного режима на другой поддерживается процессом Advisor, который реализует общий мониторинг процесса управления, работая параллельно с автоматическим контроллером AutoControl.

Решение о переходе на ручное управление принимает оператор на основе данных визуализации процесса управления. Функцией контроллера Advisor является оценка степени

расхождения между плановым и фактическим функционированием объекта с определением желательности перехода на ручное управление. Отметим, что фиксация аварийного или катастрофического состояния должна быть упреждающей с тем, чтобы оператор имел время проанализировать угрожающую ситуацию для проведения адекватной модификации управления. Прогностический характер оценки процесса управления предполагает анализ истории функционирования объекта на определенном промежутке времени.

Одной из функций процесса Advisor является обеспечение безопасности от постороннего вмешательства в процесс управления применением современных методов криптографии, т.е. аутентификации, шифрования посылаемой через Интернет информации и обеспечение ее целостности (защиту от модификации).

Окружение. Запуск контроллера системы управления реализуется сообщением init(passw, task), где passw – пароль для доступа в систему управления, task – исходное задание на процесс функционирования объекта управления. Сообщение show(viz) передает дополнительные данные viz для визуализации оператору результатов анализа процесса управления контроллером Advisor.

Состояние. Набор переменных состояния s тот же самый, что и для процесса AutoControl.

Управляющие состояния: a0, a1, a2.

Требования.

a0, init(passw, task) → Launch(passw, task**:** s #a1**:** "wrong password" #a0)
a1 → start, a2
a2 → Monitoring(s**:** s', viz), show(viz)

В управляющем состоянии a0 контроллер ожидает сообщения init(passw, task), которое генерируется при начальном запуске системы управления оператором. При получении сообщения контроллер запускает программу-гиперфункцию Launch, которая проверяет правильность пароля passw, набранного оператором и переданного контроллеру в зашифрованном виде. Если пароль passw не совпадает с хранящимся в контроллере, то выдается сообщение оператору о неправильном пароле с предложением ввести его снова, и управление возвращается в состояние a0. При правильном пароле контроллер переносит информацию о задании task в переменные состояния s и переходит в управляющее состояние a1.

В управляющем состоянии a1 контроллер посылает сообщение start процессу AutoControl, тем самым переводя его из спящего состояние в рабочее, и переходит в управляющее состояние a2, в котором реализуется мониторинг системы управления.

Программа Monitoring(s: s', viz) сравнивает плановое и фактическое функционирование объекта управления, оценивает удовлетворение существующих ограничений и определяет возможность аварийных сценариев. Результат работы viz поставляет данные для визуализации оператору в дополнении к существующей визуализации процесса управления.

Программа. **process** Advisor(…) {

```
a0:  if (init(passw, task)) Launch(passw, task: s #a1: "wrong password"#a0)
     else #a0
a1:  start;
a2:  Monitoring(s: s', viz);
     show(viz);
     #a2
}
```

Определенная выше типовая архитектура системы управления является простейшей. Рост сложности возможен по нескольким направлениям: числу управляющих состояний, набору компонент системы, степени их параллельного взаимодействия. В системах реального времени запуск программы Step обычно реализуется независимой подзадачей. Детальное описание полного набора требований для реальных больших сложных систем занимает несколько сот страниц.

3. Постановка задачи управления квадрокоптером

3.1. Беспилотные летательные аппараты

Беспилотный летательный аппарат (БПЛА) - летательный аппарат без экипажа на борту. Управляется либо автономной бортовой системой, либо человеком дистанционно. Типы аппаратов: самолет, вертолет или ракета.

Области применения БПЛА: аэрофотосъемка, разведка, боевое применение, охрана сельхозугодий, картография, дистанционный химико-физический анализ, контроль всхожести и спелости урожая, химическая обработка, исследования птиц и животных, исследование стратосферы, работа в радиационных и токсичных зонах, поиск выживших, доставка лекарств в труднодоступные места и др. Производители БПЛА в России: Вега, Текнол, Zala, Иркут, Транзас, Аэрокон, Новик-XXI век и др.

Входные параметры (параметры in в обозначениях разд. 2.2) определяют положение и ориентацию аппарата: координаты x, y, z центра масс аппарата в неподвижной декартовой системе координат; вектор скорости v_x, v_y, v_z; вектор ускорения a_x, a_y, a_z; углы Эйлера (Тэйта-Брайана, корабельные углы) ψ, φ, θ; угловые скорости ψ', φ', θ'; угловые ускорения ψ'', φ'', θ''. С аппаратом жестко

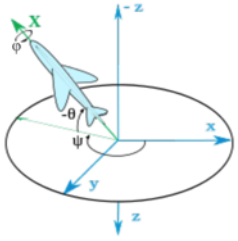

Рисунок 1 – углы Эйлера (Тэйта-Брайана)

связана локальная система координат (X, Y, Z). Ее начало совпадает с центром масс аппарата. Ось X направлена вдоль продольной оси, Y – вдоль поперечной, Z – вдоль вертикальной. Чтобы перейти из неподвижной системы координат в локальную, нужно сначала сделать поворот вокруг оси z на *угол рысканья* ψ $(-\infty;+\infty)$, потом вокруг оси Y на *угол тангажа* θ $(-\pi/2; \pi/2)$, и, наконец, вокруг оси X на *угол крена* φ $(-\pi/2; \pi/2)$.

Для определения положения и ориентации аппарата используется малогабаритная инерциальная интегрированная система, имеющая в своем составе инерциальные датчики (микромеханический гироскоп и акселерометр), а также барометрический высотомер и трехосный магнитометр. Интегрируя данные этих датчиков с данными приемника GPS / ГЛОНАСС, система вырабатывает полное навигационное решение по координатам и углам ориентации.

Управляющие параметры (com в обозначениях разд. 2.2) определяют команду и ее параметры для установки требуемых показателей ориентации и скорости БПЛА.

3.1.1. Квадрокоптеры

Мультикоптер – вертолет с количеством несущих винтов более двух. Мультикоптеры с количеством винтов больше 4 используются, как правило, для повышения мощности, а также для отказоустойчивости – такие аппараты могут продолжить движение либо совершить мягкую посадку в случае отказа нескольких винтов.

Квадрокоптеры – мультикоптеры с четырьмя несущими винтами – получили широкое распространение в качестве гражданских легких недорогих аппаратов. В них обычно используются простые винты с *постоянным шагом* – углом наклона лопастей. Каждый винт управляется собственным двигателем. Два винта вращаются по часовой стрелке, два других – против часовой. Повороты реализуются изменением скорости вращения винтов, что ограничивает маневренность из-за инерции. Квадрокоптеры, как и другие вертолеты, способны двигаться в любую сторону, зависать на месте, осуществлять вертикальный взлет и посадку.

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 проводят исследования оптимального управления экстремальных маневров, крутых виражей [27], управлением группой квадрокоптеров. Исследуются экспериментальные модели с изменяемым шагом винтов [20], обладающих лучшей маневренностью.

Входные параметры квадрокоптера AR.DRONE: x, y, z, ψ, φ, θ, v_{xb}, v_{yb}. Вместо v_x, v_y используются v_{xb}, v_{yb} – проекции вектора скорости (v_x, v_y, v_z) на локальные оси X, Y. Остальные параметры те же, что и для БПЛА. Аппарат 200 раз в секунду передает параметры на устройство дистанционного управления. Значения координат x, y, z поступают от внешних камер, либо вычисляются бортовой инерциальной системой [26] или с помощью компьютерного зрения [28].

Управляющие параметры квадрокоптера AR.DRONE: φ, θ, ψ', v_z. Значения этих параметров задаются в ограниченных диапазонах. Управляющие параметры трансформируются в команду управления квадрокоптером. Аппарат может принимать команды управления каждые 30мс.

3.2. Задача управления полетом квадрокоптера по заданному маршруту

Маршрут полета квадрокоптера определяется набором *контрольных точек* $P_1, P_1, \ldots, P_n$ в пространстве координат x, y, z; где P_1 – начальная точка маршрута, а P_n – конечная точка. На первом этапе перед началом полета строится *траектория полета* – непрерывная кривая $T(u)=(x(u), y(u), z(u))$, $u\in[0,1]$, проходящая по контрольным точкам, причем $T(0)=P_1$ и $T(1)=P_n$. В случае, когда для каждой контрольной точки дополнительно фиксируется время ее прохождения, траектория принимает вид: $T(t)=(x(t), y(t), z(t))$, $t\in[0,t_{max}]$. Простейшая траектория – ломаная линия не оптимальна по времени, поскольку в контрольных точках аппарат будет вынужден полностью останавливаться. Оптимальные по времени траектории – гладкие кривые, построенные с учетом маневренности квадрокоптера.

Построение траектории полета обычно совмещается с планированием полета; например, определяется скорость движения на разных участках маршрута. В критических ситуациях построенная траектория может корректироваться в процессе полета [21].

Задача *управления полетом по траектории* часто ставится при наличии ограничений различного рода. Это могут быть как общие ограничения по времени, скорости, угловой скорости, высоте и т.д; так и ограничения в контрольных точках по времени, скорости [12] и ориентации аппарата. Более серьезными являются пространственные ограничения. Их нарушения оказываются фатальными. Требуется аккуратное планирование, гарантирующее полет на достаточном удалении от препятствий с запасом на возможные возмущения. Неподвижные препятствия могут задаваться в виде трехмерных объектов [12, 15, 24, 31]. Иногда фиксируется допустимый коридор отклонения траектории от ломаной, соединяющей контрольные точки [9]. Подвижные препятствия могут быть представлены трехмерными

объектами с заданным законом движения [9]. В общем случае положение подвижного объекта необходимо непрерывно отслеживать в процессе полета в целях корректировки траектории полета.

Дополнительные трудности возникают при длительной потере сигнала со спутников, обслуживающих приемники GPS / ГЛОНАСС, когда при навигации лишь через инерциальные датчики [26] постепенно накапливаются ошибки в определении координат местоположения. Актуальной становится задача автоматического ориентирования в пространстве с использованием средств компьютерного зрения [28].

На очередной шаге управления полетом программа Step по входным параметрам in (координатам, ориентации, скорости и угловой скорости квадрокоптера) вычисляет управляющие параметры com в целях изменения скорости и угловой скорости для достижения целевых параметров par (координат квадрокоптера через определенное время). В качестве par может выступать одна из точек на траектории полета. Возможны разные стратегии выбора такой точки как вблизи квадрокоптера, так и на достаточно большом удалении от него. Входные параметры in посылаются квадрокоптером в некоторый момент времени t_0, а управляющие команды com срабатывают на квадрокоптере в другой момент времени t_1, когда его параметры уже другие. В принципе, значение параметров во время t_1 можно было бы вычислить, используя уравнения движения квадрокоптера [9, 15, 24], однако на практике такое редко реализуется, что порождает т.н. ошибку инерции. Сопоставляя фактические координаты с рассчитанными для момента времени t_1 и запоминая эту информацию в параметрах состояния s можно эффективно прогнозировать характер возмущений в целях внесения соответствующих компенсирующих поправок.

3.3. Задача слежения за движущимся объектом

Задача слежения за движущимся объектом заключается в сопровождении объекта квадрокоптером на фиксированном расстоянии от него [7, 16, 23, 30]. Предполагается, что объект перемещается по земле. К квадрокоптеру прикреплена видео камера на вращающемся подвесе. Задача – реализовать управление квадрокоптером и подвесом, чтобы объект всегда был в кадре видеокамеры. Варианты слежения:

1. «Вид от третьего лица»: следование за объектом сверху-сзади. Параметры: расстояние до объекта R, относительная высота H.
2. «Вид сбоку»: следование за объектом со стороны. Параметры: направление по компасу A, расстояние R, относительная высота H.

3. «Вращение»: квадрокоптер вращается вокруг объекта. Параметры: период вращения T, расстояние R, относительная высота H.

Слежение завершается:

- при достижении заданного местоположения;
- после истечения заданного времени полета;
- после получения команды с пульта управления.

По завершении слежения квадрокоптер должен мягко сесть на землю.

Развернутая спецификация задачи слежения должна учитывать множество аспектов. Некоторые из них перечислены ниже.

Модель камеры и подвеса:

- Углы обзора кадра: горизонтальный $F\psi$ и вертикальный $F\theta$.
- Углы ориентации подвеса: $G\psi$, $G\varphi$, $G\theta$.

Подсистема локации объекта. Положение объекта относительно квадрокоптера определяется параметрами: направлением и расстоянием. Подсистема локации может быть реализована одним из способов:

- в виде системы компьютерного зрения;
- ближний радар и радиомаяк (например, RFID);
- объект имеет высокоточный GPS приемник.

Предсказание траектории объекта. Необходимо предсказывать траекторию объекта. Подвижность традиционных объектов видеосъемки – автомобилей, мотоциклов, танков и другой наземной техники, лыжников, сноубордистов, велосипедистов, а также ускорения и угловые ускорения объекта – ограничены.

Несмотря на то, что объект передвигается по земле, он может совершать прыжки в несколько метров (лыжник, мотоциклист). Также следует учитывать перепады высоты ландшафта.

Аварийные случаи. Система управления должна уметь определить аварийную ситуацию и предпринять меры для безопасного прекращения работы. Примеры аварийных ситуаций: потеря связи, сильный ветер, низкий заряд батареи, техническая неисправность. Безопасное прекращение работы обычно включает в себя мягкую посадку в определенной зоне, уведомление оператора о неисправности и местоположении аппарата.

Система управления квадрокоптером соединяет в себе элементы жесткого и мягкого реального времени. Так как траектория движения объекта заранее не известна, система управления должна планировать и корректировать траекторию движения квадрокоптера и

подвеса камеры на несколько секунд вперед. При этом необходимо учитывать препятствия – горы, деревья, здания.

Качество работы системы управления определяется параметрами:

- минимальным отклонением от целевых параметров (расстояние до объекта, высота и тд);
- нахождение объекта как можно ближе к центру кадра;
- плавность траектории БПЛА;
- плавность перемещения объекта в полученном видео;
- правильный «горизонт» в кадре.

4. Обзор работ

Интеграция автоматического и ручного управления в стиле human-in-the-loop [19] реализована в автомобильной, авиационной и космической отраслях. Определены пять уровней автоматизации вождения автомобиля от ручного до полностью автоматического [22]. Переход на тотальное автоматическое управление всего потока автомобилей планируется с 2020г. в рамках грандиозных программ США и Европы по развитию автомобильной отрасли [11, 22]. Отметим, что ручное и автоматическое управление квадракоптеров реализовано независимо. Их интеграция изначально не планировалась. В работах по системам управления квадрокоптерами просматриваются лишь отдельные элементы такой интеграции.

Движение квадрокоптера описывает плоская система дифференциальных уравнений [9, 15, 24], позволяющая заменить нелинейную систему уравнений линеаризованной системой в окрестности положения равновесия [12, 17, 26]. Для линейной системы строится асимптотически устойчивое решение с ограничением на угол крена. При больших углах крена линеаризованная модель дает значительную ошибку.

Нелинейные системы решаются методами обратного шага (back-stepping) нелинейного программирования [9, 10, 18, 25].

Для построения траектории для исходного маршрута полета строится гладкая кривая, заданная полиномиальными сплайнами. Далее траектория параметризуется временем так, чтобы траектория была осуществима, а общее время полета стало минимальным [12, 15, 31].

Планирование оптимальных траекторий с учетом препятствий реализуется методом графа видимости [12], быстрым маршевым методом [12] или алгоритмом RRT* [24].

Исследования по управлению квадрокоптером AR.DRONE проводились в Институте автоматики и электрометрии СО РАН (г. Новосибирск) [1, 2]. Построена модель бортовой

системы управления, система управления и измерения положения в пространстве, метод фильтрации шумов с помощью фильтра Калмана.

Слежение за подвижными наземными целями с помощью бортовых видеокамер БПЛА активно изучается [7, 16, 23, 30]. В 2015 г. планируется выпуск коммерческих квадрокоптеров для личного пользования с функцией видеосъемки и слежения за объектом «AirDog» и «Hexo+».

В мировой практике технология определения требований разрабатывается и применяется в основном для больших интернетовских информационных систем и систем телекоммуникации, а также в системной инженерии (системотехнике). Технология спецификации требований отражена в стандарте [13]. В нашем подходе определение требований рассматривается для всего класса реактивных систем.

5. Заключение

Разработка программного обеспечения, реализующего системы управления БПЛА в различных приложениях, становится все более популярной отраслью; см., например, [5]. Актуальной становится не только разработка отдельных методов управления БПЛА, но и исследование постановки задачи управления БПЛА в целом с анализом всего множества разнообразных аспектов, таких как интеграция автоматического и ручного управления или обеспечение безопасности от постороннего вмешательства.

Настоящая работа возникла в рамках исследований по технологии автоматного программирования [4]. Цель работы – сформулировать задачу построения произвольной системы управления на примере задачи управления квадракоптером. Данный подход предопределил метод построения типовой архитектуры системы управления с последующей ее специализацией для задач управления БПЛА. Анализ методов построения для других подклассов систем управления, особенно для автомобильной отрасли, позволил обнаружить архитектуру систем управления вида human-in-the-loop [19] как базисную для многих подклассов систем управления, в т.ч. и для систем управления квадракоптерами, несмотря на то, что в настоящее время не наблюдается интеграции автоматического и ручного управления квадрокоптерами. Однако есть свидетельства того, что такая интеграция скоро состоится. Для этого в частности необходимо будет разработать методы предсказания поведения БПЛА на базе анализа предыдущей истории. Сходные методы будут также применимы и для компенсации возмущений.

Список литературы

1. Белоконь С.А., Золотухин Ю.Н., Мальцев А.С., Нестеров А.А., Филиппов М.Н., Ян А.П. Управление параметрами полёта квадрокоптера при движении по заданной траектории // Автометрия, 2012. С. 32-42.
2. Белоконь С.А., Золотухин Ю.Н., Котов К.Ю., Мальцев А.С., Нестеров А.А., Филиппов М.Н., Ян А.П. Управление квадрокоптером AR.DRONE при движении по заданной траектории // Проблемы управления и моделирования в сложных системах: Труды XV международной конференции, 25-28 июня 2013 г., Самара, С. 506-514.
3. Новиков Д.А. Теория управления организационными системами / МПСИ, Москва. 2005. 584 с.
4. Шелехов В.И. Разработка автоматных программ на базе определения требований // Системная информатика, №4, 2014. — ИСИ СО РАН, Новосибирск. — С. 1-29. http://persons.iis.nsk.su/files/persons/pages/req_tech.pdf
5. “Хозяин, напиши для нас приложение”. Требуется разработчик софта и железа для дронов DJI. // Блог “Хакспейс Neuron” на сайте habrahabr.ru. URL: http://habrahabr.ru/company/neuronspace/blog/259527/ (дата обращения: 20.08.2015).
6. A. Cockburn. Writing effective use cases / Addison-Wesley. 2001. 270 P.
7. C. Teuliere, L. Eck, and E. Marchand. Chasing a moving target from a flying uav. // IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011.
8. D. Klahr, P. Langley, R. Neches. Production system models of learning and development. Cambridge, Mass.: The MIT Press. 1987. 467 P.
9. D. Mellinger, V. Kumar. Minimum snap trajectory generation and control for quadrotors // Robotics and Automation (ICRA), 2011 IEEE International Conf.
10. 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.
11. European Roadmap Smart Systems for Automated Driving. Version 1.2. Berlin, April 1st, 2015, 39 P.
12. G.M. Hoffman, S.L. Waslander, C.J. Tomlin. Quadrotor helicopter trajectory tracking control // In proc. AIAA Guidance, Navigation and Control Conf, 2008.
13. IEEE recommended practice for software requirements specifications. Revision: 29/Dec/11.
14. I.D. Cowling, J.F. Whidborne, A.K. Cooke. Optimal trajectory planning and LQR control for a quadrotor UAV. // in International Control Conference, 2006.

15. I.D. Cowling, O.A. Yakimenko, J.F. Whidborne. A prototype of an autonomous controller for a quadrotor UAV // In European Control Conference, 2007.
16. J. Xiao, C. Yang, F. Han, H. Cheng, and Sarnoff Corporation. Vehicle and person tracking in aerial videos. // Multimodal Technologies for Perception of Humans, 2008. P. 203–214.
17. K. Miller. Path tracking control for quadrotor helicopters, 2008.
18. L. Lai, C. Yang, C. Wu. Time-optimal control of a hovering quad-rotor helicopter // Journal of Intelligent and Robotic Systems, 45(2), June 2006. P. 115-135.
19. Li W., Sadigh D., Sastry S., Seshia S. Synthesis for human-in-the-loop control systems // Tools and Algorithms for the Construction and Analysis of Systems. LNCS 8413, 2014, P. 470–484.
20. M. Cutler, J.P. How. Actuator constrained trajectory generation and control for variable-pitch quadrotors, 2012.
21. M. Hehn, R. D'Andrea. Quadrocopter trajectory generation and control // IFAC World Congress, 2011, P. 1485-1491.
22. National Highway Traffic Safety Administration. Preliminary statement of policy concerning automated vehicles. May 2013, 14 P.
23. P. Theodorakopoulos and S. Lacroix. Uav target tracking using an adversarial iterative prediction. // IEEE International Conference on Robotics and Automation, 2009.
24. R. Charles, A. Bry, N. Roy. Polynomial trajectory planning for quadrotor flight // Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, 2013.
25. R. Cunha, D. Cabecinhas, C. Silvestre. Nonlinear trajectory tracking control of a quadrotor vehicle. // In Proc. European Control Conference, 2009.
26. R.W. Beard. Quadrotor dynamics and control // Brigham Young University, Feb 2008
27. S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea. A simple learning strategy for high-speed quadro-copter multi-flips. // IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2010. P. 1642–1648.
28. 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.
29. S.H. Yang, L.S. Tan, and X. Chen. Requirements specification and architecture design for Internet-based control systems // Computer Software and Applications Conference, 2002. Proceedings. 26th Annual International, 26-29 Aug. 2002. P. 75-80.
30. S. Zhang. Object tracking in unmanned aerial vehicle (uav) videos using a combined approach // IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.

31. Y. Bouktir, M. Haddad, T. Chettibi. Trajectory planning for a quadrotor helicopter // In Mediterranean Conference on Control and Automation. IEEE, June 2008.