

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И СИСТЕМ

**Любич Леонард Яковлевич**

**Выпускная квалификационная работа бакалавра**

**Автоматическое управление посадкой  
квадрокоптера на движущееся основание**

Направление 010400

Прикладная математика и информатика

Научный руководитель,  
кандидат физ.-мат. наук,  
доцент  
Сотникова М. В.

Санкт-Петербург

2016

# Содержание

Введение .....	3
Глава 1. Постановка задачи .....	5
1.1. Общая схема миссии автопосадки .....	6
1.2. Моделирование динамики квадрокоптера .....	9
1.3. Система компьютерного зрения .....	13
1.4. Математическая постановка задачи .....	16
Глава 2. Разработка алгоритма управления .....	18
2.1. Алгоритм Backstepping .....	19
2.2. Общая схема алгоритма Backstepping .....	20
2.3. Применение алгоритма Backstepping к квадрокоптеру .....	22
Глава 3. Программная реализация алгоритма управления .....	25
3.1. Алгоритм Backstepping .....	25
3.2. Трёхмерная имитация системы компьютерного зрения .....	28
3.3. Результаты имитационного моделирования .....	31
Выводы .....	34
Список литературы .....	35

## Введение

Развитие систем с использованием беспилотных летательных аппаратов (БЛА) является современной тенденцией в аэрокосмических исследованиях. Беспилотным летательным аппаратом называется движущийся объект, на борту которого в процессе полета отсутствует человек. Такие БЛА могут пилотироваться дистанционно или автономно следовать по предварительно запрограммированному пути. БЛА в основном используются в военных операциях. Однако они находят применение и в гражданских нуждах, таких как пожаротушение. В целом, беспилотные аппараты предпочтительны при выполнении неподходящих для пилотируемых аппаратов миссий.

Существует широкий спектр геометрических конфигураций, средств управления и общего функционирования БЛА. Аппараты с лопастными моторами имеют ряд преимуществ перед вариантами с фиксированными крыльями. К ним в первую очередь относятся способность вертикально взлетать и садиться, а также возможность парить над неподвижной точкой. Кроме того, БЛА вертолетного типа могут перемещаться в диапазоне низких скоростей.

Одну из возможных конфигураций моторов вертолета представляет так называемый квадрокоптер – летательный аппарат с 4-мя основными винтами и отсутствующим хвостовым. Все несущие винты расположены в одной плоскости. Управление движением квадрокоптера достигается за счет изменения скоростей вращения каждого пропеллера.



Рис. 1. Квадрокоптер

Целью данной работы является реализация управления посадкой квадрокоптера на движущуюся платформу. Движение платформы происходит в горизонтальной плоскости, что может соответствовать участку земли или другому подвижному объекту с очень малыми колебаниями по вертикали. Основным и единственным координатором движения является сам БЛА, оценивающий обстановку по видеоданным, получаемым с бортовой камеры.

Квадрокоптер представляет из себя транспортное средство с четырьмя входными силами, которые преобразуются в шесть управляющих воздействий. Существует ряд способов управления движением такой системы. Могут применяться методы управления линейными системами (PID-контроллер, LQR). Однако, поскольку квадрокоптер является нелинейной системой, для должной функциональности необходимо применение нелинейных способов управления. Наиболее употребимыми являются методы Backstepping, Sliding и метод линеаризации обратной связи.

В работе используется закон управления Backstepping, примененный к квадрокоптеру для разработки надежной системы стабилизации и преследования траектории.

Нелинейный Backstepping позволяет построить закон управления для следования желаемой траектории. В рамках работы таковой является движение целевой точки посадки. Траектория платформы изначально считается неопределенной. Считается, что посадочная область двигается с небольшой скоростью и ее траектория является непрерывно дифференцируемой, поэтому в течение малых промежутков времени её можно приближать линейной. Это учитывается при симуляции миссии полета.

В работе описана математическая модель квадрокоптера, система компьютерного зрения, построена общая схема полета при выполнении миссии посадки и синтезирован закон управления для нее. Также представлены результаты имитационного моделирования тестовых полетов.

# Глава 1. Постановка задачи

Пусть имеется один квадрокоптер с симметрично расположенными винтами и жестко прикрепленной ко дну видеокамерой. Также есть круговая плоская платформа, которая двигается в горизонтальной плоскости в пределах обнаружения ее квадрокоптером. Радиус платформы незначительно превосходит расстояние от центра масс до края любого из винтов. На стартовом этапе квадрокоптер находится на земле, моторы выключены. Целевым является состояние, когда квадрокоптер сел на движущуюся платформу.

Требуется:

- построить математическую модель динамики квадрокоптера в виде системы обыкновенных дифференциальных уравнений второго порядка;
- выбрать алгоритм обнаружения по видеоданным, поступающим с бортовой камеры, текущего положения платформы относительно квадрокоптера;
- построить закон управления, который обеспечивает стабилизацию аппарата и следованию за траекторией движения платформы;
- построить алгоритм перехода квадрокоптера из начального состояния в область посадки;
- реализовать программно симуляцию этого перехода;
- получить и проанализировать результаты, сделать выводы.

Предполагается, что квадрокоптер не может работать в штормовых условиях: сильном дожде, порывистом ветре, высокой турбулентности.

## 1.1. Общая схема миссии автопосадки

Для летательных аппаратов углы поворота вокруг осей прямоугольной системы координат приняты следующие названия: угол крена (roll), угол тангажа (pitch) и угол рыскания (yaw).

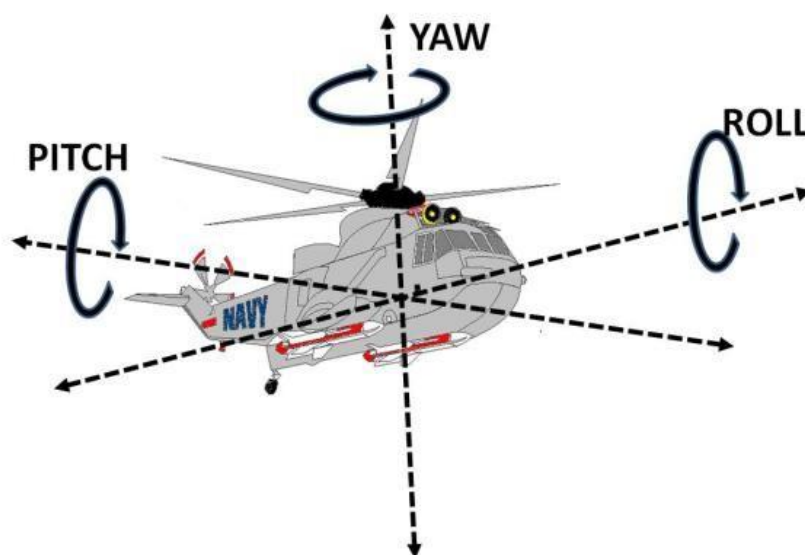


Рис. 2. Крен, тангаж и рыскание.

Потребуем выполнения следующих условий при посадке [1]:

- Крен и тангаж в момент посадки должны быть нулевыми

$$\theta^{des} = 0, \quad \varphi^{des} = 0.$$

- Точка посадки должна быть в центре платформы

$$x = x_c^{plat}, \quad y = y_c^{plat}.$$

- На завершающем этапе вертикальная скорость квадрокоптера относительно платформы должна подчиняться условию

$$\dot{z} \leq 0.5 \text{ м/с}.$$

Единственное, чем может руководствоваться автопилотная система, – получаемые с бортовой камеры кадры. По ним система должна быть способна вычислить свое положение относительно платформы. Такой подход в иностранной литературе называется image-based [1]. Главный минус подхода заключается в том, что потеря цели на кадре лишает квадрокоптер

возможности оценить свое положение относительно платформы. Для компенсации этого недостатка обычно добавляется GPS-навигация.

В работе предполагается, что квадрокоптер изначально находится в небольшой удаленности от платформы.

Ниже представлена блок-схема режимов контроллера во время полета и их детальное описание.

### **Режим поиска**

В этом режиме квадрокоптер, начиная полет из начального состояния, набирает высоту до тех пор, пока платформа не будет обнаружена на кадре, полученном с камеры. Предполагается, что цель будет двигаться в пределах некоторой ограниченной области, не выходя за ее пределы. Обозначим характерный радиус этой области через  $R_p$ . Пусть FOV (field of view) – угол обзора камеры. Тогда высота, с которой цель может быть гарантированно обнаружена, вычисляется по формуле

$$z_{search} = \frac{R_p}{tg(\min\{FOV\}/2)} \quad (1)$$

К каждому получаемому с камеры кадру применяется алгоритм обнаружения области платформы.

### **Режим отслеживания**

После того, как цель была обнаружена видеокамерой, квадрокоптер начинает выполнять задачу позиционирования над платформой на заданной высоте.

К каждому кадру по-прежнему применяется алгоритм обнаружения. В случае потери цели контроллер возвращается в режим поиска.

### **Режим сближения**

После выполнения задачи позиционирования квадрокоптер выравнивается с центром платформы. Это способствует уменьшению вероятности расхождения с ней во время финального снижения.

### **Режим посадки**

Данный режим управляет вертикальной скоростью квадрокоптера,

который должен приземлиться как можно ближе к заданной точке посадки.

При сближении БЛА (следовательно, и камеры) с целью, ее изображение полностью заполняет кадр. В следствие этого, система видеонаблюдения оказывается не в состоянии определить положение квадрокоптера относительно платформы. Данная проблема может быть решена изменением внешнего вида платформы с добавлением дополнительных ориентиров. В данной работе этот вопрос не рассматривается.

При разработке программного комплекса имитационного моделирования в среде MATLAB-Simulink математическая модель, регулятор и платформа реализуются в виде отдельных подсистем, связанных друг с другом. При этом режимы полета выбираются в процессе моделирования в зависимости от текущего состояния системы.

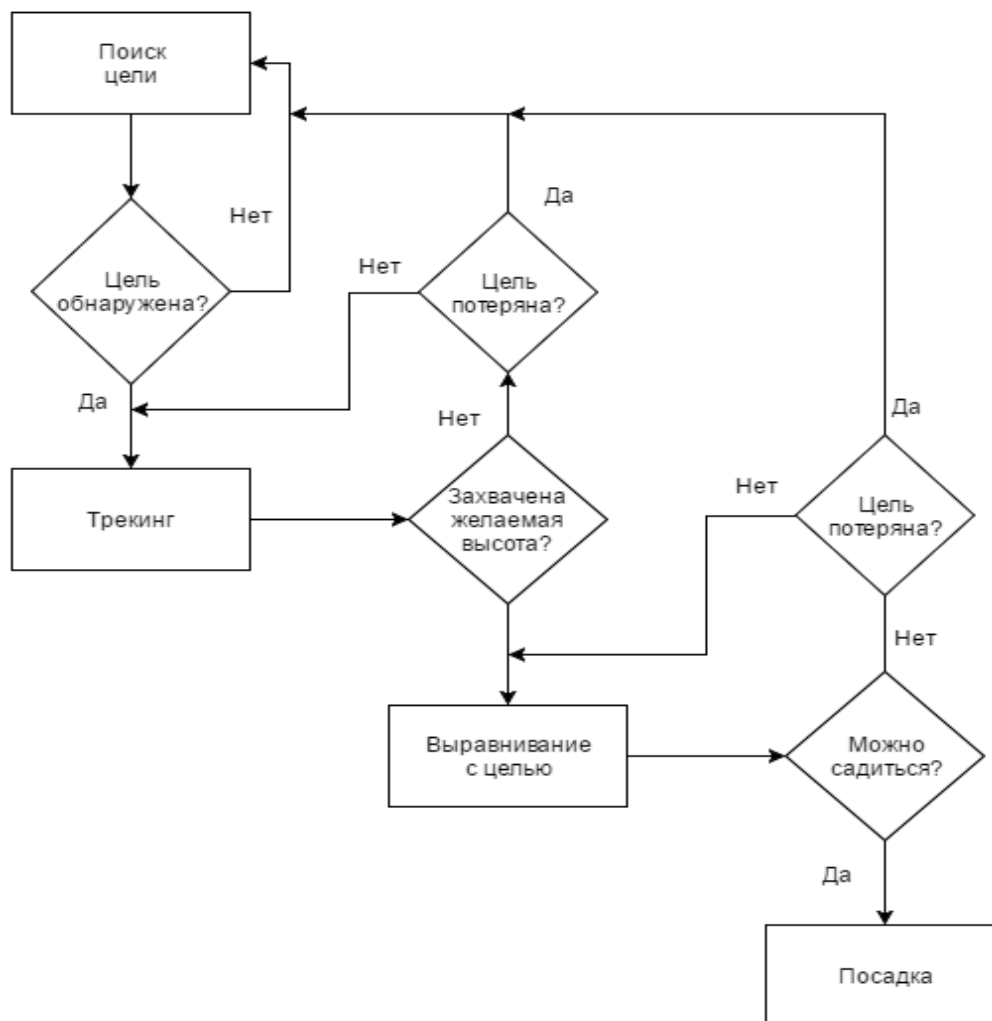


Рис. 3. Схема миссии.



Особенность подхода image-based control состоит в том, что целевое состояние системы изначально выбирается как желаемый образ цели на кадре, а впоследствии преобразуется в желаемое положение квадрокоптера.

## 1.2. Моделирование динамики квадрокоптера

Математическая модель квадрокоптера необходима для разработки регуляторов, а также для их тестирования в имитационных полетах.

Используемые предположения [1]:

- квадрокоптер имеет жесткую структуру;
- квадрокоптер симметричен;
- осевая нагрузка моторов направлена вертикально;
- масса постоянна.

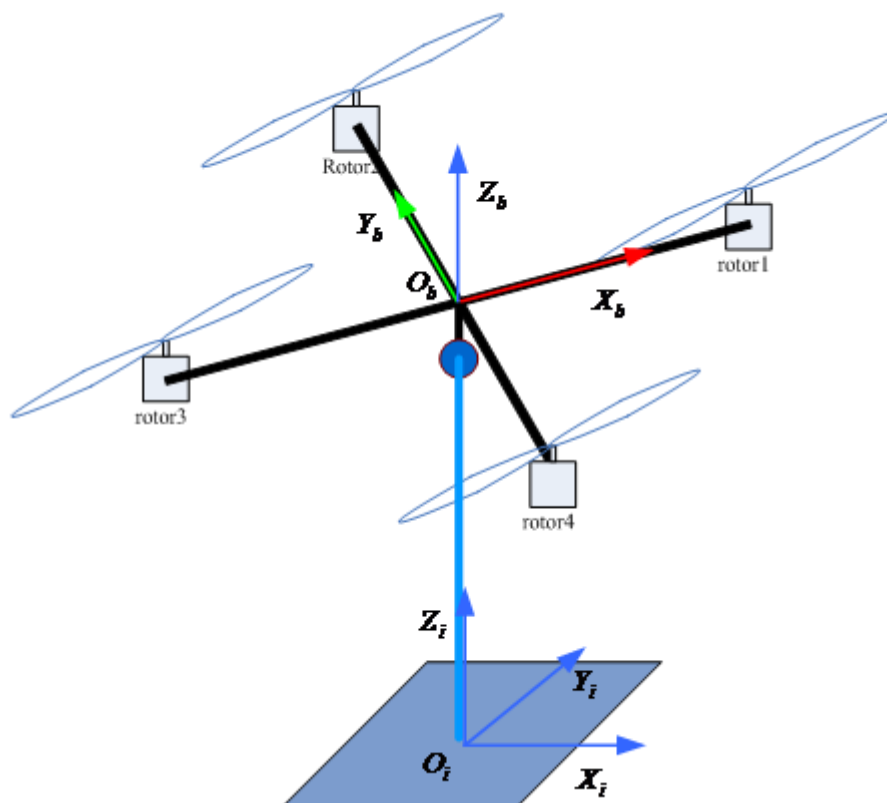


Рис. 4. Системы координат.

Будем использовать две системы координат:

1. связанная с Землей ортогональная система координат  $O_i x_i y_i z_i$ .  
Ось  $O_i x_i$  направлена на восток, ось  $O_i y_i$  на север, ось  $O_i z_i$

дополняет до правой тройки;

- связанная с телом квадрокоптера ортогональная система координат  $O_b x_b y_b z_b$ . Ось  $O_b x_b$  направлена вправо, ось  $O_b y_b$  вперед, ось  $O_b z_b$  дополняет до правой тройки.

Введем следующие обозначения:

- положение:  $\bar{P} = (x, y, z)^T$ ;
- скорость:  $\bar{V} = (u, v, w)^T$ ;
- углы Эйлера:  $\bar{\theta} = (\varphi, \theta, \psi)^T$ ;
- угловые скорости:  $\bar{\omega} = (p, q, r)^T$ .

Переход от системы координат  $O_b x_b y_b z_b$  к системе координат  $O_i x_i y_i z_i$  осуществляется при помощи матрицы поворота [3]:

$$T^{IB} = \begin{bmatrix} C_\psi C_\theta & S_\psi C_\theta & -S_\theta \\ C_\psi S_\theta S_\varphi - S_\psi C_\varphi & S_\psi S_\theta S_\varphi + C_\psi C_\varphi & C_\theta S_\varphi \\ C_\psi S_\theta C_\varphi + S_\psi S_\varphi & S_\psi S_\theta C_\varphi - C_\psi S_\varphi & C_\theta C_\varphi \end{bmatrix}, \quad (2)$$

где  $C_\gamma = \cos(\gamma)$ ,  $S_\gamma = \sin(\gamma)$ .

Обратный переход осуществляется матрицей  $T^{BI} = T^{IB^{-1}} = T^{IB^T}$ .

Запишем кинематические дифференциальные уравнения для углов Эйлера через угловые скорости [1]:

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\varphi \tan\theta & \cos\varphi \tan\theta \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi / \cos\theta & \cos\varphi / \cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \quad (3)$$

где  $\varphi, \theta, \psi$  – крен, тангаж и рыскание соответственно.

Из второго закона Ньютона могут быть получены уравнения движения квадрокоптера [1]:

$$\bar{F} = \frac{d}{dt}(m\bar{V}), \quad (4)$$

$$\bar{M} = \frac{d\bar{H}}{dt}. \quad (5)$$

Распишем уравнения (4), (5) в скалярной форме:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} -mg + m\ddot{u} \\ m\dot{v} \\ m\dot{w} \end{bmatrix}, \quad (6)$$

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} I_x\dot{p} + (I_z - I_y)qr \\ I_y\dot{q} + (I_x - I_z)rp \\ I_z\dot{r} + (I_y - I_x)pq \end{bmatrix}, \quad (7)$$

где  $F_i, M_i$  – это проекции сил и моментов сил (уравнения записаны в системе координат  $O_b x_b y_b z_b$ ).

Совместное управление  $U_1$  есть суммарная подъемная сила, вызванная четырьмя моторами одновременно. Вращение по углу тангажа возникает при возрастании (падении) скорости заднего мотора, тогда как скорость переднего убывает (растет). Движение по крену получается ускорением (замедлением) правого мотора совместно с замедлением (ускорением) левого. Угол рыскания изменяется, когда одновременно растут (падают) скорости вращения переднего и заднего моторов, а боковых падают (растут).

Динамическая модель квадрокоптера [9] в виде системы ОДУ имеет вид:

$$\left\{ \begin{array}{l} \ddot{\phi} = \frac{1}{I_x} [(I_z - I_y)qr + M_x + M_{x_d}] \\ \ddot{\theta} = \frac{1}{I_y} [(I_z - I_x)rp + M_y + M_{y_d}] \\ \ddot{\psi} = \frac{1}{I_z} [(I_x - I_y)pq + M_z + M_{z_d}] \\ \ddot{x} = \frac{F_x}{m} \\ \ddot{y} = \frac{F_y}{m} \\ \ddot{z} = -g + \frac{F_z}{m} \end{array} \right. \quad (8)$$

Здесь  $M_i$  – это моменты управляемых сил,  $M_{i_d}$  – моменты внешних возмущений,  $F_i$  – управляемые силы,  $F_{i_d}$  – силы внешних возмущений,  $m$  – масса квадрокоптера,  $l$  – расстояние от центра масс до винта. Эти величины вычисляются по формулам:

$$M_x = \frac{l}{I_x} U_2, M_y = \frac{l}{I_y} U_3, M_z = \frac{l}{I_z} U_4,$$

$$M_{x_d} = -\frac{J_r}{I_x} q \Omega, M_{y_d} = -\frac{J_r}{I_y} p \Omega, M_{z_d} = 0, \quad (9)$$

$$F_x = (\cos \psi \sin \theta \cos \varphi + \sin \psi \sin \varphi) U_1,$$

$$F_y = (\sin \psi \sin \theta \cos \varphi - \sin \psi \sin \varphi) U_1,$$

$$F_z = (\cos \varphi \cos \theta) U_1. \quad (10)$$

Управляющие сигналы  $U_1, U_2, U_3, U_4$  и возмущение  $\Omega$  находятся через скорости моторов по следующим формулам:

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2),$$

$$U_2 = b(-\Omega_2^2 + \Omega_4^2),$$

$$U_3 = b(-\Omega_1^2 + \Omega_3^2), \quad (11)$$

$$U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2),$$

$$\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4.$$

Динамическую модель, описанную системой (8), можно переписать в виде ЛТП-модели в ss-форме. Определим вектор состояния:

$$X^T = [\varphi \quad \dot{\varphi} \quad \theta \quad \dot{\theta} \quad \psi \quad \dot{\psi} \quad z \quad \dot{z} \quad x \quad \dot{x} \quad y \quad \dot{y}],$$

$$x_1 = \varphi, x_2 = \dot{\varphi} = \dot{x}_1, x_3 = \theta, x_4 = \dot{\theta} = \dot{x}_3, x_5 = \psi, x_6 = \dot{\psi} = \dot{x}_5, \quad (12)$$

$$x_7 = z, x_8 = \dot{z}, x_9 = x, x_{10} = \dot{x}, x_{11} = y, x_{12} = \dot{y}.$$

$$\dot{X} = f(X, u) = \begin{pmatrix} x_2 \\ x_4 x_6 a_1 + x_4 a_2 \Omega + b_1 U_2 \\ x_4 \\ x_2 x_6 a_3 + x_2 a_4 \Omega + b_2 U_3 \\ x_6 \\ x_4 x_6 a_5 + b_3 U_4 \\ x_8 \\ -g + (\cos x_1 \cos x_3) \frac{1}{m} U_1 \\ x_{10} \\ u_x \frac{1}{m} U_1 \\ x_{12} \\ u_y \frac{1}{m} U_1 \end{pmatrix}, \quad (13)$$

где

$$a_1 = (I_y - I_z)/I_x, a_2 = -J_r/I_x, a_3 = (I_z - I_x)/I_y, a_4 = -J_r/I_y,$$

$$a_5 = (I_x - I_y)/I_z, b_1 = l/I_x, b_2 = l/I_y, b_3 = l/I_z,$$

$$u_x = (\cos x_1 \sin x_3 \cos x_5 + \sin x_1 \sin x_5),$$

$$u_y = (\sin x_1 \sin x_3 \cos x_5 - \sin x_1 \sin x_5).$$

### 1.3. Система компьютерного зрения

Бортовая камера является неподвижной и направлена вертикально вниз. Она используется для определения положения квадрокоптера относительно платформы.

Характеристики цели могут быть разнообразны и многочисленны. Геометрическая конструкция и другие свойства цели существенно влияют на сложность алгоритма обнаружения, поэтому целесообразно определить их заранее. Можно выделить следующие основные требования:

- предпочтительны простые геометрические формы (круги или квадраты);
- размеры: с одной стороны целевая граница должна быть достаточно малой, чтобы помещаться в кадр изображения при сближении квадрокоптера с платформой; с другой стороны, при слишком малых размерах цель будет занимать недостаточное количество пикселей для качественной работы алгоритмов компьютерного зрения;
- цвет платформы следует выбрать таким, чтобы он не существовал в изобилии в природе (например, красный).

Из вышеуказанных характеристик очевидно, что на изображении камеры цель будет отображаться как некоторое красное пятно. Требуется разработать соответствующий алгоритм обработки изображений, который обнаруживает и выявляет свойства красных пятен на кадре, состоящий из следующих этапов:

1. поиск красных пикселей;
2. конвертация изображения в бинарное;
3. извлечение свойств пятен.

### **Поиск красных пикселей**

Обнаружение объекта производится путем сканирования всех пикселей захваченного изображения. При использовании цветовой модели RGB каждый пиксель представлен 3х-элементным вектором. «Чистому» красному пикселю будет соответствовать значение [255, 0, 0]. В реальной ситуации такие значения почти не встречаются, поэтому нужно допустить определенное отклонение и включить в рассмотрение «красноподобные» пиксели. Ниже представлены некоторые варианты критерия «красноты» [1]:

- $\sqrt{(R - 255)^2 + G^2 + B^2} < \varepsilon_1$
- $R > \varepsilon_2 \cdot G \ \&\& \ R > \varepsilon_2 \cdot B$
- $R > \varepsilon_3 \cdot (G + B)$

### **Конвертация в бинарное изображение**

Черный цвет присваивается пикселям, которые по какому-либо из критериев приняты красными, все остальные принимаются белыми. Путем таких преобразований платформа на изображении примет вид сплошной черной кляксы.

### **Извлечение свойств пятен**

Важно отметить, что вполне возможно множественное появление черных клякс на кадрах. Необходимо построить фильтр для исключения шумовых пятен. Затем, учитывая что нужные сгустки пикселей обнаружены, можно вычислить некоторые их свойства: площадь, периметр, центр, координаты краев.

По кадру на изображении можно вычислить позицию соответствующей пикселю точки относительно камеры. Введя плоскую ортогональную систему координат на изображении, центр которой находится в центре кадра, порядковый номер каждого пикселя можно принять за его координаты в этой

системе. По расстоянию между пикселями  $d_p$  и известному реальному расстоянию  $d_{real}$  можно вычислить высоту, с которой захвачено изображение:

$$z = \frac{s_c f d_{real}}{d_p}, \quad (15)$$

где  $f$  это фокусное расстояние,  $s_c$  это масштабирующий множитель.

Относительное горизонтальное положение точки вычисляется как

$$x = \frac{z(x_p - x_o)}{s_c f}, y = \frac{z(y_p - y_o)}{s_c f}. \quad (16)$$

Координаты центра пятна вычисляются по формулам:

$$\begin{aligned} x_{target} &= (\min(circle_x) + \max(circle_x))/2, \\ y_{target} &= (\min(circle_y) + \max(circle_y))/2. \end{aligned} \quad (17)$$

Для фильтрации пятен можно ввести два критерия [1]:

- занимаемая область на изображении должна быть не менее, чем пороговое значение  $\varepsilon_{Area}$ ;
- показатель округлости области  $\left| \frac{\text{периметр}}{2\pi} - \frac{\text{диаметр}}{2} \right|$  должен быть меньше  $\varepsilon_R$ .

Заканчивая обзор системы зрения, отдельно подчеркнем, что координатная плоскость камеры в системе координат тела имеет собственную ориентацию, что требуется учитывать при составлении отношений между точкой на изображении и точкой в пространстве. Это должно быть учтено при составлении отношений между точкой на изображении и точкой в пространстве. Матрица перехода имеет вид:

$$T^{CI} = T^{CB} \cdot T^{BI}. \quad (18)$$

Построенная система зрения позволяет выявлять траекторию перемещения центра платформы в пространстве, руководствуясь только видеоданными с камеры. Следование этой траектории является целью управления в процессе полета квадрокоптера.

Как видно на рис. 5, на кадре обнаружено множество черных элементов

разного размера. С помощью представленного выше подхода возможно выделить самую крупную область, наиболее похожую на цель.

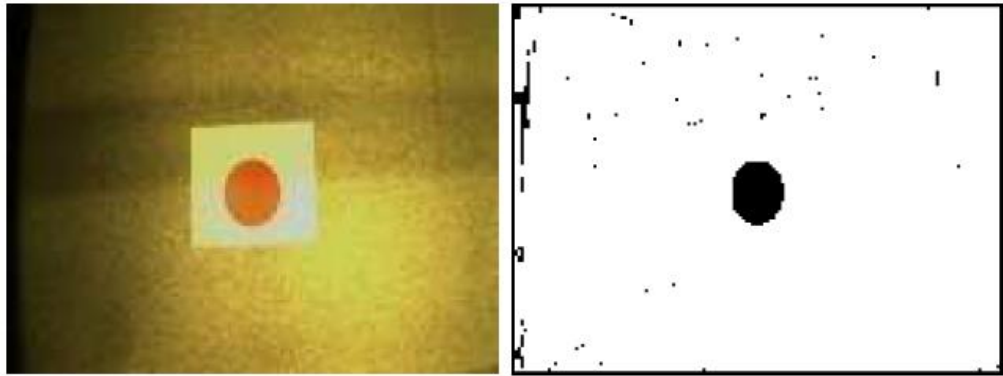


Рис.5. Пример работы алгоритма обнаружения красных пикселей.

### 1.4. Математическая постановка задачи

Пусть круговая платформа с центром  $(x_c, y_c)$  движется по некоторому закону:

$$\begin{cases} x = x_{plat}(t) \\ y = y_{plat}(t), \\ z = 0 \end{cases}$$

а перемещение квадрокоптера в пространстве задается изменением его координат  $(x, y, z) = [x(t), y(t), z(t)]$ .

Динамика квадрокоптера описывается системой (16).

Вектор управления определяется как  $u = (U_1, U_2, U_3, U_4)^T$ . Все компоненты зависят от настраиваемых параметров. Следовательно, обозначив набор параметров через  $K$ , управление становится функцией этих параметров:  $u = u(K)$ .

Требуется построить управление  $u = u(K)$  для системы (16):

$$\begin{cases} \lim_{t \rightarrow \infty} |x - x_{plat}(t)| = 0 \\ \lim_{t \rightarrow \infty} |y - y_{plat}(t)| = 0' \end{cases}$$

и обеспечить выполнение условия

$$z = 0 \Rightarrow \begin{cases} |x - x_c| < \Delta \\ |y - y_c| < \Delta' \end{cases}$$

где  $\Delta$  обозначает разницу между радиусом платформы и радиусом



окружности, описывающей квадрокоптер.

Основные идеи и принципы построения решения могут быть почерпнуты в [1]. Также в этом источнике глубоко рассматривается image-based подход.

Управление квадрокоптером является сложной задачей из-за его нелинейного поведения. В [2] описывается моделирование и управление квадрокоптером.

В [3] автор производит сравнение линейных и нелинейных методов управления квадрокоптерами.

Нелинейный метод управления Backstepping довольно подробно описывается [4].

Управление летательным аппаратом с 4 винтами при помощи компьютерного зрения исследуется в работе [5].

Вопрос навигации летательных аппаратов с использованием данных, получаемых с бортовой камеры описан в [6].

Для управления квадрокоптером существует целый ряд различных методов. Их можно подразделить на линейные, такие как LQR и PID, рассматриваемые в [7], и нелинейные, в частности Backstepping и Sliding [9].

В [8] автор дает общую схему синтеза управления квадрокоптером.

## Глава 2. Разработка алгоритма управления

Квадрокоптер является неустойчивой системой, поэтому для управления таким аппаратом требуется контроллер с обратной связью. Backstepping – подход к синтезу регуляторов для нелинейных систем, основанный на теории устойчивости Ляпунова [4]. Метод разработан в 1990г. П. Кокотовичем для построения алгоритмов управления системами определенной структуры. Эти системы имеют рекурсивную структуру, т.е. состоят из отдельных подсистем, на вход каждой из которых поступают данные из стабильной внешней по отношению к ней подсистемы.

Построенная в предыдущей главе математическая модель квадрокоптера (1) имеет вышеописанную структуру. Первые три ОДУ представляют подсистему, отвечающую за вращение БЛА в пространстве. В свою очередь следующие три уравнения описывают горизонтальные перемещения [8]. Причем вторая подсистема зависит от выходного состояния первой. Следовательно, применительно к построенной модели backstepping начинает работу со стабилизации по углам и угловым скоростям, результаты которой будут использованы при дальнейшей стабилизации по позиции и скоростям.

Верно построенный закон управления Backstepping обеспечивает асимптотическую сходимость вектора состояния системы к желаемой величине. Применительно к данной работе это позволит квадрокоптеру совершать следование за центром платформы с целью максимального сближения с ней и дальнейшей посадки. Иначе говоря, кроме стабилизации движения самого квадрокоптера, аппарат, используя методы компьютерного зрения, самостоятельно определит целевую траекторию и будет следовать по ней.

## 2.1. Метод управления Backstepping

Рассмотрим систему следующего вида:

$$\left\{ \begin{array}{l} \dot{x} = f_x(x) + g_x(x)z_1 \\ \dot{z}_1 = f_1(x, z_1) + g_1(x, z_1)z_2 \\ \vdots \\ \dot{z}_i = f_i(x, z_1, z_2, \dots, z_{i-1}, z_i) + g_i(x, z_1, z_2, \dots, z_{i-1}, z_i)z_{i+1} \\ \vdots \\ \dot{z}_k = f_k(x, z_1, z_2, \dots, z_{k-1}, z_k) + g_k(x, z_1, z_2, \dots, z_{k-1}, z_k)u \end{array} \right. \quad (19)$$

где  $x \in R^n, z_i \in R^1, u \in R^1$  - вход в систему,  $f_i(\bar{0}) = 0$ .

При построении управления для системы (19) будем пользоваться теоремой Ляпунова об устойчивости.

**Теорема Ляпунова об асимптотической устойчивости.** Пусть имеется система  $\frac{dx}{dt} = f(x, t)$ . При этом считается, что множество  $Z = \{(t, x) \mid t \geq 0, \|x\| \leq H\}$  является областью существования и единственности решения задачи Коши. Если для системы существует положительно определенная функция Ляпунова  $V(t, x)$ , непрерывно дифференцируемая по совокупности своих аргументов, допускающая бесконечно малый высший предел и имеющая отрицательно определенную производную  $\dot{V}(t, x)$  по времени в силу системы, то нулевое решение  $x = \bar{0}$  рассматриваемой системы асимптотически устойчиво по Ляпунову.

Пусть первая в (19) подсистема  $\dot{x} = f_x(x) + g_x(x)u_x(x)$  стабилизирована некоторым известным управлением  $u_x(x): u_x(0) = 0$ . Пусть также известна функция Ляпунова  $V_x$  для стабилизированной системы.

Управление  $u$ , построенное с помощью Backstepping, даст непосредственно стабилизирующее воздействие на состояние  $z_k$ . В свою очередь состояние  $z_k$  действует как стабилизирующее управление для состояния  $z_{k-1}$ . Это процесс продолжается аналогичным образом, и каждое состояние  $z_i$  стабилизируется фиктивным управлением  $z_{i+1}$ . Проход в обратном направлении от управления  $u$  до управления  $z_1$  послужил причиной названия метода Backstepping, т.е. «шагание назад».

Таким образом, подход Backstepping состоит в определении того, как стабилизировать подсистему  $x$  с помощью выхода подсистемы  $z_1$ , приводом которой является  $z_2$ , и так далее. Процесс «спускается вниз» до тех пор, пока не будет построено последнее управление  $u$ . Пошаговая схема:

1.  $\dot{x} = f_x(x) + g_x(x)u_x(x)$  стабилизирована, функция Ляпунова  $V_x$ ;
2. Управление  $u_1(x, z_1)$  строится так, что  $z_1$  сходится к  $u_x$ . При построении используется  $V_1(x, z_1) = V_x(x) + \frac{1}{2}(z_1 - u_x(x))^2$ ;
3. Управление  $u_2(x, z_1, z_2)$  строится так, что  $z_2$  сходится к  $u_1$ . При построении используется  $V_2(x, z_1, z_2) = V_1(x, z_1) + \frac{1}{2}(z_2 - u_1(x, z_1))^2$ ;
4. Далее все аналогично, пока не будет построено последнее управление  $u$ .

## 2.2. Общая схема алгоритма Backstepping

### Одношаговая процедура

Рассмотрим простую систему:

$$\begin{cases} \dot{x} = f_x(x) + g_x(x)z_1 \\ \dot{z}_1 = f_1(x, z_1) + g_1(x, z_1)u_1 \end{cases} \quad (20)$$

Вместо того, чтобы проектировать управление с обратной связью  $u_1$ , вводится новое управление  $u_{a1}$  и используется новый закон:

$$u_1(x, z_1) = \frac{1}{g_1(x, z_1)}(u_{a1} - f_1(x, z_1)). \quad (21)$$

Перепишем систему (20) после подстановки управления (21):

$$\begin{cases} \dot{x} = f_x(x) + g_x(x)z_1 \\ \dot{z}_1 = f_1(x, z_1) + g_1(x, z_1) \frac{1}{g_1(x, z_1)}(u_{a1} - f_1(x, z_1))' \end{cases} \quad (22)$$

$$\begin{cases} \dot{x} = f_x(x) + g_x(x)z_1 \\ \dot{z}_1 = u_{a1} \end{cases}. \quad (23)$$

Считая известными закон управления  $u_x(x)$  и функцию Ляпунова  $V_x(x)$

для первой подсистемы (23), получим управление для второй подсистемы:

$$u_{a1}(x, z_1) = -\frac{\partial V_x}{\partial x} g_x(x) - k_1(z_1 - u_x(x)) + \frac{\partial u_x}{\partial x}(f_x(x) + g_x(x)z_1), \quad (24)$$

где  $k_1 > 0$ .

Итоговый стабилизирующий закон управления:

$$u_1(x, z_1) = \frac{1}{g_1(x, z_1)} \left( -\frac{\partial V_x}{\partial x} g_x(x) - k_1(z_1 - u_x(x)) + \frac{\partial u_x}{\partial x}(f_x(x) + g_x(x)z_1) - f_1(x, z_1) \right). \quad (25)$$

Соответствующая функция Ляпунова:

$$V_1(x, z_1) = V_x(x) + \frac{1}{2}(z_1 - u_x(x))^2. \quad (26)$$

Таким образом, исходная система (20) теперь имеет известные стабилизирующее управление и функцию Ляпунова для него. По методу Backstepping теперь эта система может рассматриваться как часть более сложной системы.

### Многошаговая процедура

Исходная система имеет рекурсивную структуру:

$$\left\{ \left\{ \left\{ \begin{array}{l} \dot{x} = f_x(x) + g_x(x)z_1 \\ \dot{z}_1 = f_1(x, z_1) + g_1(x, z_1)u_1 \\ \dot{z}_2 = f_2(x, z_1, z_2) + g_2(x, z_1, z_2)z_3 \\ \vdots \\ \dot{z}_i = f_i(x, z_1, z_2, \dots, z_{i-1}, z_i) + g_i(x, z_1, z_2, \dots, z_{i-1}, z_i)z_{i+1} \\ \vdots \\ \dot{z}_{k-1} = f_{k-1}(x, z_1, z_2, \dots, z_{k-2}, z_{k-1}) + g_{k-1}(x, z_1, z_2, \dots, z_{k-2}, z_{k-1})z_k \\ \dot{z}_k = f_k(x, z_1, z_2, \dots, z_{k-1}, z_k) + g_k(x, z_1, z_2, \dots, z_{k-1}, z_k)u \end{array} \right. \right. \right. \quad (27)$$

и может быть стабилизирована обратной связью нахождением управления и функции Ляпунова для интегратора подсистемы  $(x, z_1)$  в (27), после чего процесс выходит из этой внутренней подсистемы и продолжается, пока не будет получен стабилизирующий закон управления  $u$ .

На  $i$ -ом шаге система (27) эквивалентна:

$$\begin{cases} \dot{x} \\ \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_{i-2} \\ \dot{z}_{i-1} \end{cases} \stackrel{\triangle}{=} \begin{cases} \dot{x}_{i-1} \\ \dot{x} \\ \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_{i-2} \\ \dot{z}_{i-1} \end{cases} = \begin{bmatrix} \overbrace{f_{i-2}(x_{i-2}) + g_{i-2}(x_{i-2})z_{i-2}}^{\triangle f_{i-1}(x_{i-1})} \\ f_{i-1}(x_i) \end{bmatrix} + \begin{bmatrix} \overbrace{0}^{\triangle g_{i-1}(x_{i-1})} \\ g_{i-1}(x_i) \end{bmatrix}. \quad (28)$$

$$\dot{z}_i = f_i(x_i) + g_i(x_i)u_i$$

Управление:

$$\begin{aligned} & u_i(\overbrace{x, z_1, \dots, z_i}^{\triangle x_i}) \\ &= \frac{1}{g_i(x_i)} \left[ -\frac{\partial V_{i-1}}{\partial x_{i-1}} g_{i-1}(x_{i-1}) - k_i(z_i - u_{i-1}(x_{i-1})) \right. \\ & \left. + \frac{\partial u_{i-1}}{\partial x_{i-1}} (f_{i-1}(x_{i-1}) + g_{i-1}(x_{i-1})z_i) - f_i(x_{i-1}) \right], \end{aligned} \quad (29)$$

где  $k_i > 0$ .

Функция Ляпунова:

$$V_i(x_i) = V_{i-1}(x_{i-1}) + \frac{1}{2}(z_i - u_{i-1}(x_{i-1}))^2. \quad (30)$$

На шаге  $i = k$  может быть получено конечное управление

$$u(x, z_1, z_2, \dots, z_k) = u_k(x_k).$$

### 2.3. Применение алгоритма Backstepping к квадрокоптеру

Математическая модель квадрокоптера описывается системой обыкновенных дифференциальных уравнений (8). В ней первые три уравнения описывают подсистему угловых вращений, а последующие три – горизонтальных перемещений. Будем работать с системой, представленной в state-space форме (13). Для начала рассмотрим входное управление для вращательной подсистемы, из которой в дальнейшем выведем управление координатами.

Пусть  $X = X_d$  — желаемое состояние модели, которое считается целевой траекторией движения квадрокоптера. Построим управление, обеспечивающее следование по этой траектории системой.

Для первого уравнения системы (13) введем ошибку преследования:

$$z_1 = x_{1d} - x_1 \quad (31)$$

и функцию Ляпунова:

$$V(z_1) = \frac{1}{2} z_1^2, \quad (32)$$

производная которой имеет вид:

$$\dot{V}(z_1) = z_1(\dot{x}_{1d} - \dot{x}_1). \quad (33)$$

Представляя виртуальное входное управление  $x_2$  для стабилизации  $z_1$

$$x_2 = \dot{x}_{1d} + k_1 z_1, \quad (34)$$

где  $k_1 > 0$ , получаем производную функции Ляпунова в виде:

$$\dot{V}(z_1) = -k_1 z_1^2. \quad (35)$$

Введя обозначения:

$$\begin{aligned} z_2 &= x_2 - \dot{x}_{1d} - k_1 z_1, \\ V(z_1, z_2) &= \frac{1}{2} (z_1^2 + z_2^2). \end{aligned} \quad (36)$$

следует, что производная функции Ляпунова  $V(z_1, z_2)$  выражается:

$$\begin{aligned} \dot{V}(z_1, z_2) &= z_2(a_1 x_4 x_6 + a_2 x_4 \Omega + b_1 U_2) - z_2(\ddot{x}_{1d} - k_1(z_2 \\ &+ k_1 z_1)) - z_1 z_2 - k_1 z_1^2, \end{aligned} \quad (37)$$

и должно выполняться:  $\dot{V}(z_1, z_2) < 0$ .

Таким образом,  $U_2$  может быть вычислено:

$$U_2 = \frac{1}{b_1} (z_1 - a_1 x_4 x_6 - a_2 x_4 \Omega - b_1 U_2 - k_1(z_2 + k_1 z_1)) - k_2 z_2 \quad (38)$$

По той же процедуре вычисляются  $U_3$  и  $U_4$

$$U_3 = \frac{1}{b_2} (z_3 - a_3 x_2 x_6 - a_4 x_2 \Omega - k_3(z_4 + k_3 z_3) - k_4 z_4), \quad (39)$$

$$U_4 = \frac{1}{b_3} (z_5 - a_5 x_2 x_4 - k_5(z_2 + k_1 z_1) - k_6 z_6). \quad (40)$$

Получив входные управления (38-40) для вращательной подсистемы, можно вычислить входы  $U_1$ ,  $u_x$  и  $u_y$  для нижней подсистемы [7]:

$$U_1 = \frac{m}{\cos x_1 \cos x_3} (z_7 + g - k_7(z_8 + k_7 z_7)) - k_8 z_8, \quad (41)$$

$$u_x = \frac{m}{U_1} (z_9 - k_9(z_{10} + k_9 z_9) - k_{10} z_{10}), \quad (42)$$

$$u_y = \frac{m}{U_1} (z_{11} - k_{11}(z_{12} + k_{11} z_{11}) - k_{12} z_{12}). \quad (43)$$

Здесь

$$z_3 = x_{3d} - x_3$$

$$z_4 = x_4 - \dot{x}_{3d} - k_3 z_3$$

$$z_5 = x_{5d} - x_5$$

$$z_6 = x_6 - \dot{x}_{5d} - k_5 z_5$$

$$z_7 = x_{7d} - x_7$$

$$z_8 = x_8 - \dot{x}_{7d} - k_7 z_7$$

$$z_9 = x_{9d} - x_9$$

$$z_{10} = x_{10} - \dot{x}_{9d} - k_9 z_9$$

$$z_{11} = x_{11d} - x_{11}$$

$$z_{12} = x_{12} - \dot{x}_{11d} - k_{11} z_{11}.$$

Все коэффициенты  $(k_1, \dots, k_{12})$  положительны. За счет них происходит определение интенсивности реакции на отклонение вектора состояния от желаемой траектории.



# Глава 3. Программная реализация алгоритма управления

Для проверки разработки программного кода и вычислений использовался высокоуровневый язык и интерактивная среда для программирования и численных расчетов MATLAB. Для симуляции динамики квадрокоптера и тестирования миссии автопосадки была выбрана графическая среда имитационного моделирования Simulink.

## 3.1. Алгоритм Backstepping

В первую очередь была построена математическая модель квадрокоптера, а также проведена ее стабилизация контроллером Backstepping.

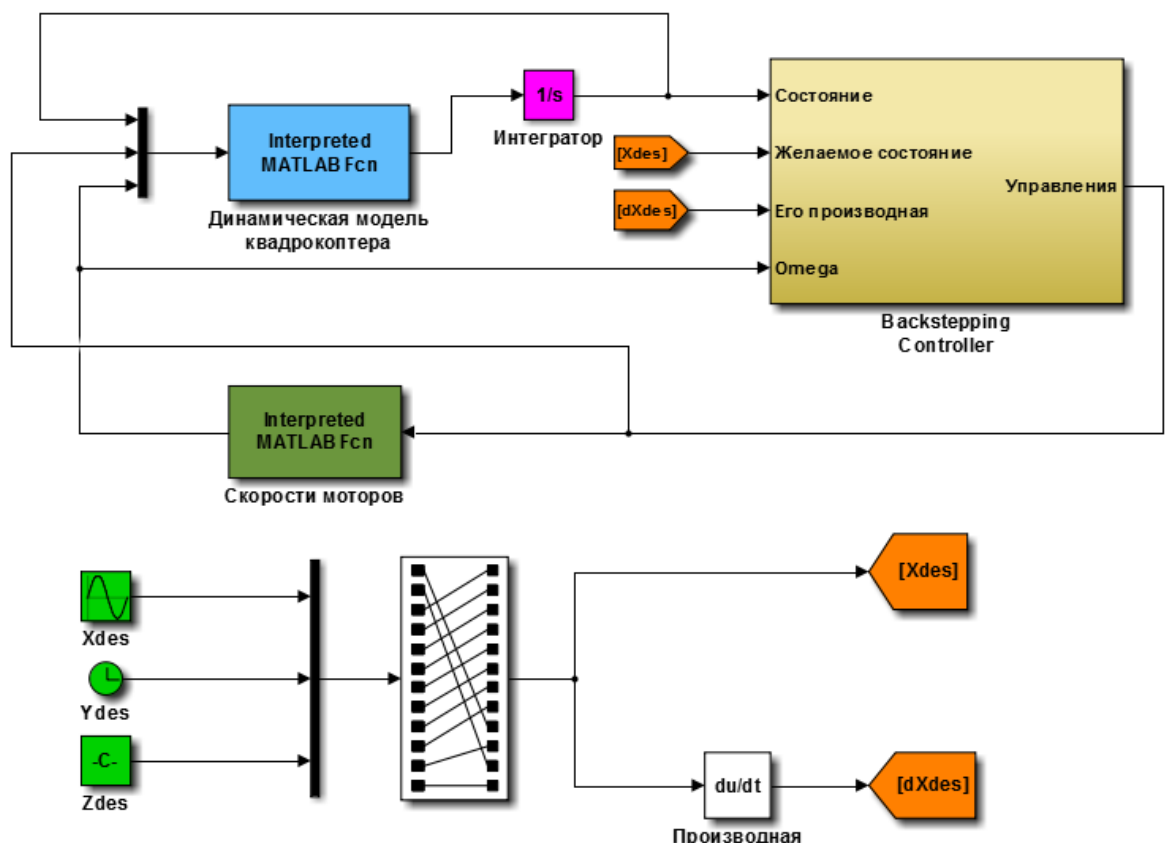


Рис.6. Модель квадрокоптера.

Блок «Динамическая модель квадрокоптера» принимает на вход вектор состояния, вектор управления и скорости вращения моторов в предыдущий

момент времени. На выходе получаем производную вектора состояния модели (13)  $\dot{X} = f(X, U)$  в текущий момент времени.

Полученный сигнал поступает на «Интегратор», а затем вместе с желаемой траекторией передается в контроллер. Подсистема Backstepping Controller преобразует входной сигнал, формируя новое управление.

Сигнал управления, проходя через блок «Скорости моторов», замыкает систему и вновь подается в модель динамики квадрокоптера для формирования нового вектора состояния. Нижний набор блоков (на рис. 6) требуется для задания желаемой траектории.

Зададим в качестве начальной позиции  $[x, y, z] = [-5, -5, 1]$ , начальной ориентации  $[\varphi, \theta, \psi] = [\pi/4, \pi/4, \pi/4]$ . Проверим работоспособность алгоритма backstepping, задав желаемый закон движения:

$$\begin{cases} x = \sin t \\ y = t \\ z = 0 \end{cases}$$

Желаемые углы крена, тангажа и рыскания нулевые.

Результат имитации отображены на последующих графиках:

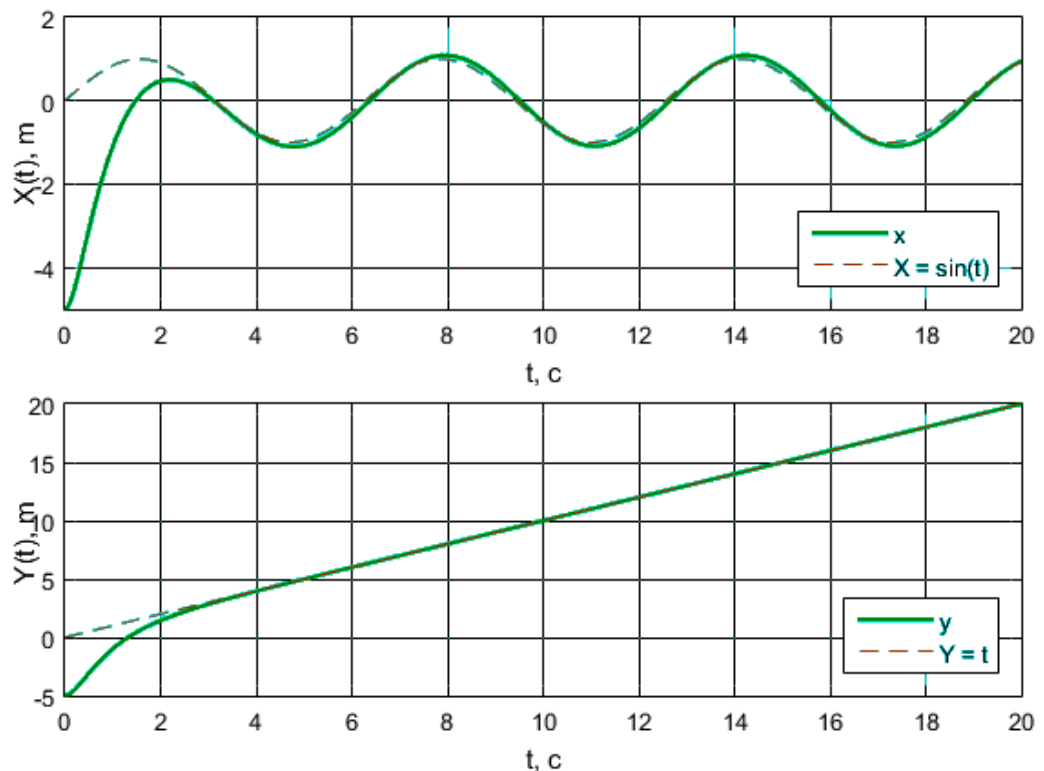


Рис.7. Графики изменения координат X и Y в период [0;20].

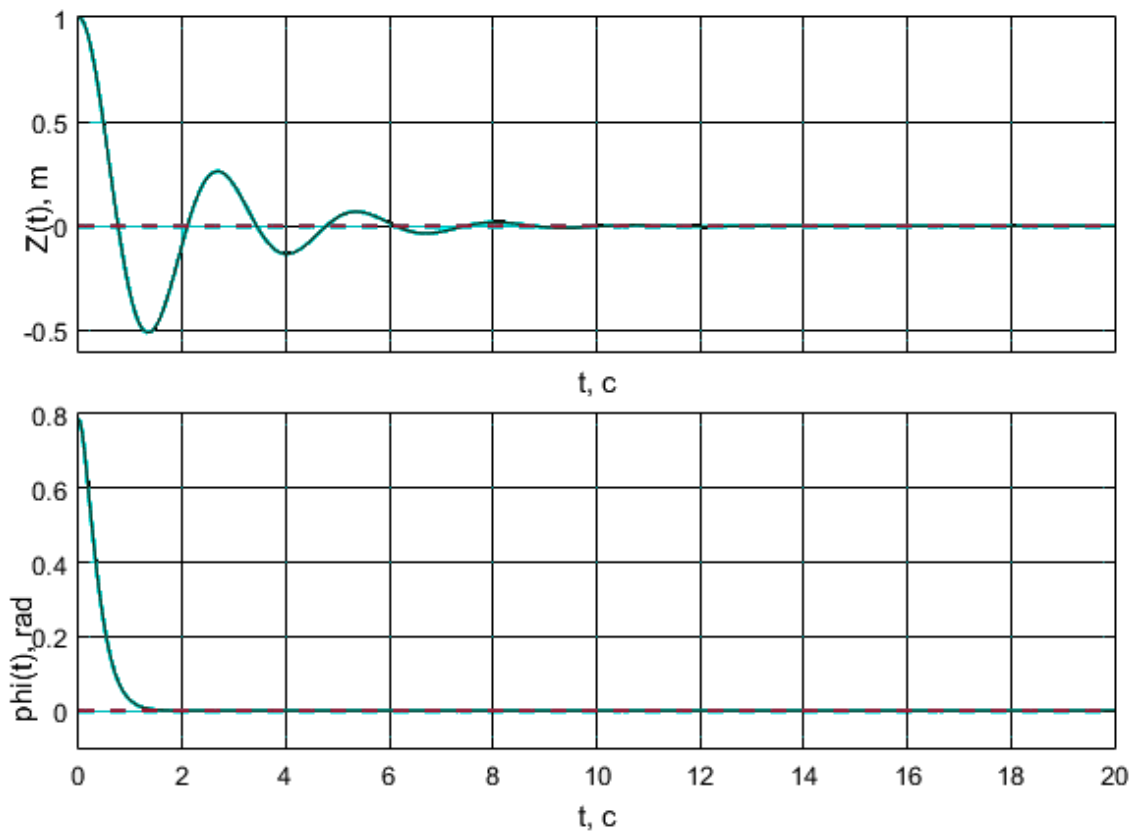


Рис. 8. Графики изменения координаты  $Z$  и угла крена в период  $[0; 20]$ .

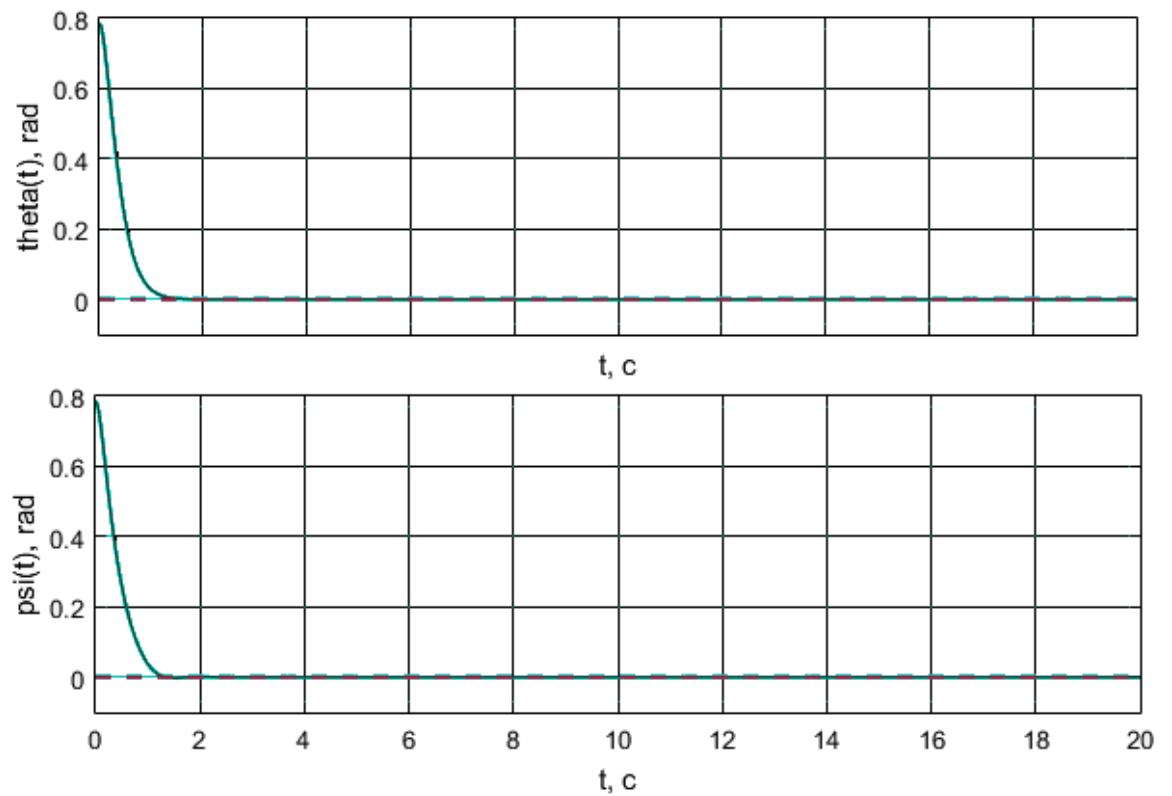


Рис. 9. Графики изменения углов тангажа и рыскания в период  $[0; 20]$ .

### 3.2. Трехмерная имитация системы компьютерного зрения

Для наглядности имитации полета и посадки квадрокоптера в среде Simulink была создана 3D-сцена. В нее помимо фона было добавлено 4 основных элемента: схематичный трехмерный макет квадрокоптера, круговая посадочная поверхность, а также две точки обзора (viewpoint, кратко VP). Первая VP соответствует бортовой камере, жесткой сцепленной с днищем летательного аппарата, вторая – стороннему наблюдателю.

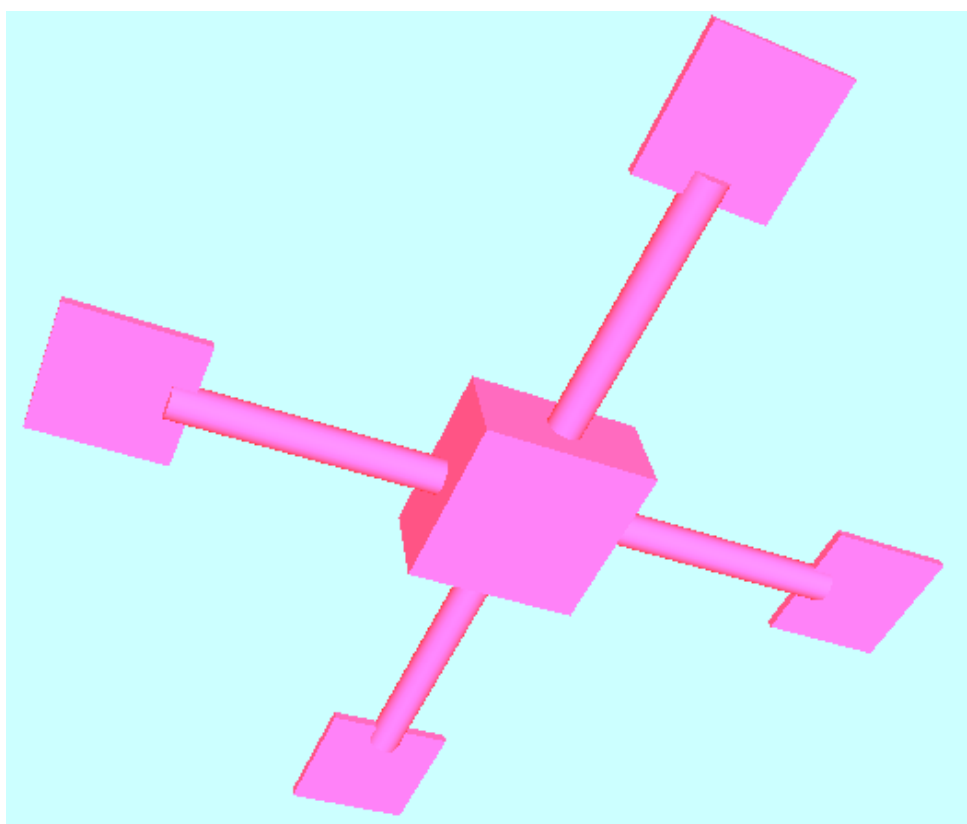


Рис. 10. Квадрокоптер, вид снизу.

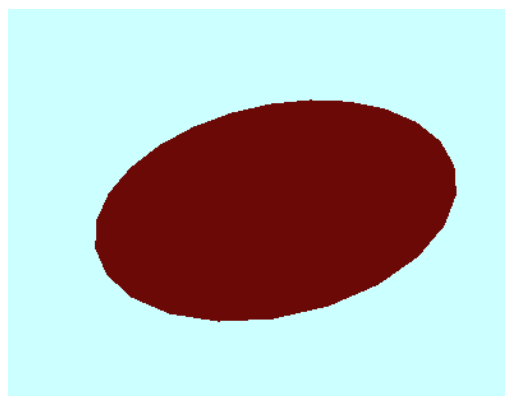


Рис.11. Платформа, вид со стороны.

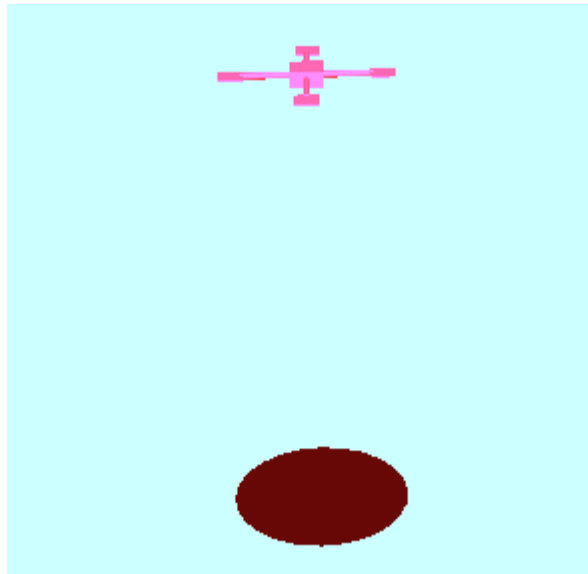


Рис. 12. Квадрокоптер парит над платформой, вид со стороны.

Составляющие модели квадрокоптера:

- 4 двигателя реализованы в виде одинаковых параллелепипедов размером  $0.15\text{м} \times 0.15\text{м} \times 0.01\text{м}$ .;
- Центральный элемент, содержащий центр масс представляет собой параллелепипед размером  $0.2\text{м} \times 0.2\text{м} \times 0.1\text{м}$ .;
- 4 связующих звена в виде одинаковых цилиндров высотой  $0.3\text{м}$  и радиусом  $0.02\text{м}$  каждый.

От платформы в 3D-сцену добавлена только верхняя посадочная поверхность в виде плоского круга радиусом  $0.6\text{м}$ .

Для построения управления в данной работе принимаются решения, основанные только на данных обработки получаемых изображений с бортовой камеры. Для захвата видео в процессе полета использовались инструменты библиотеки Simulink 3D Animation. Кадры передаются функциям библиотеки 8-битными изображениями, получаемыми с VP, прикрепленной к нижней части коптера.

Ниже показан пример части изображения и его обработанного варианта, снятого камерой с  $5\text{ метров}$  над платформой (Рис. 13). Кадр обрабатывается алгоритмом, описанным в главе 1.2.

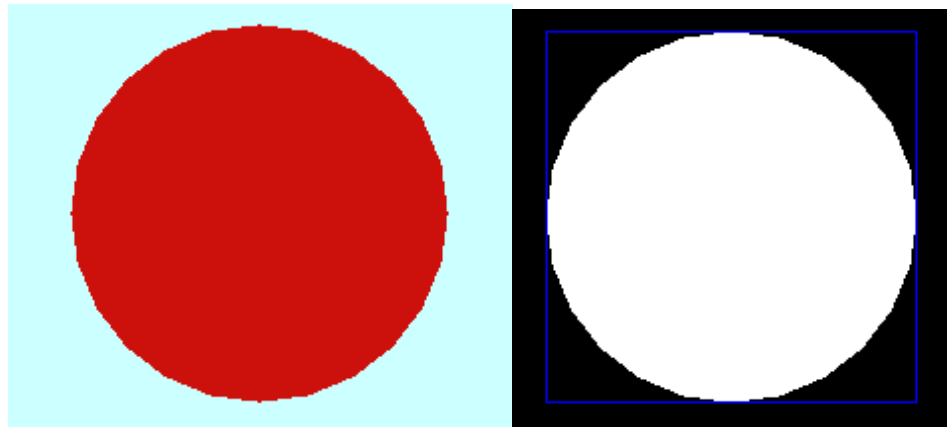


Рис. 13. Квадрокоптер обнаружил цель.

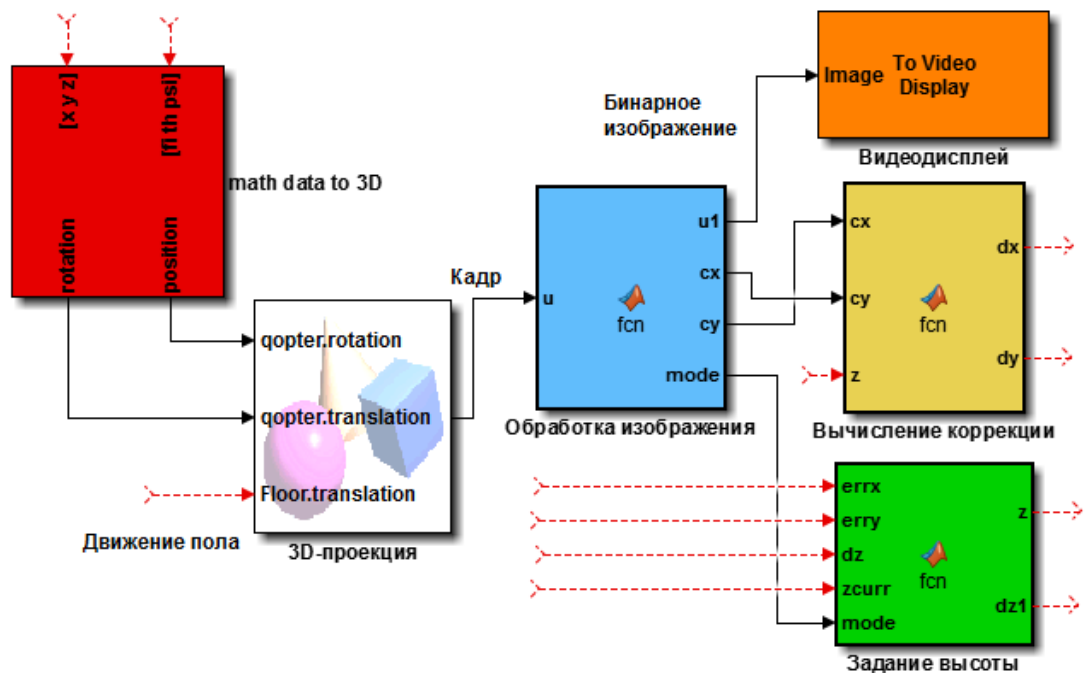


Рис. 14. Подсистема компьютерного зрения.

Simulink-модель на рис.14 строит текущее состояние системы с помощью двух блоков: «math data to 3D» и «3D-проекция». Первый преобразует координаты, второй формирует изображение бортовой камеры. Сигнал «Кадр» содержит 8-битное изображение, подаваемое в блок для его обработки. Внутри происходят вычисления, и на выходе имеются бинарное изображение, координаты центра цели на кадре, а также режим полета. Далее координаты центра платформы используются для определения горизонтальной коррекции движения.

### 3.3. Результаты имитационного моделирования

Следующие значения констант и параметров модели были выбраны при реализации программного полета [2]:

Обозначение	Название	Значение
$m$	Масса квадрокоптера	0.65 кг
$I_x$	Момент инерции отн. оси $x$	$7.5 \cdot 10^{-3}$ кг · м <sup>2</sup>
$I_y$	Момент инерции отн. оси $y$	$7.5 \cdot 10^{-3}$ кг · м <sup>2</sup>
$I_z$	Момент инерции отн. оси $z$	$1.3 \cdot 10^{-2}$ кг · м <sup>2</sup>
$b$	Коэффициент тяги	$3 \cdot 10^{-5}$ Н · с <sup>2</sup>
$d$	Коэффициент сопротивления	$7.5 \cdot 10^{-7}$ Н · м · с <sup>2</sup>
$J_r$	Инерция двигателя	$6 \cdot 10^{-5}$ кг · м <sup>2</sup>
$l$	Длина рукоятки	0.3 м

Построенный закон управления Backstepping содержит 12 числовых задаваемых параметров. Их вариация влияет на реакцию регулятора на отклонение системы от желаемого состояния, и, следовательно, на скорость сходимости.

Для симуляции полета были выбраны следующие значения:

$$k_1 = 5, k_2 = 5, k_3 = 5, k_4 = 5, k_5 = 5, k_6 = 5,$$

$$k_7 = 0.5, k_8 = 0.5, k_9 = 5, k_{10} = 1, k_{11} = 5, k_{12} = 1.$$

Стоит отметить, что выбор оптимального набора коэффициентов  $k_i$  выходит за рамки данной работы.

Квадрокоптер находится в начальной позиции:

$$x_0 = -1, y_0 = -1, z_0 = 0.$$

Платформа движется прямолинейно:

$$x_{plat}(t) = 0, y_{plat}(t) = t.$$

В результате моделирования было показано, что автопилотируемый аппарат взлетел, и приземлился на цель за 6 с секунд. Ниже представлены графики изменения положения во время полета, а также отклонение от

желаемой прямолинейной траектории.

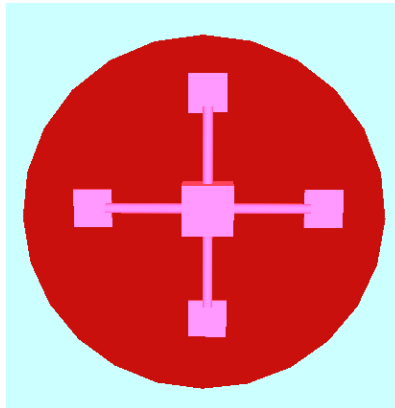


Рис.15. Квадрокоптер сел на цель, вид сверху.

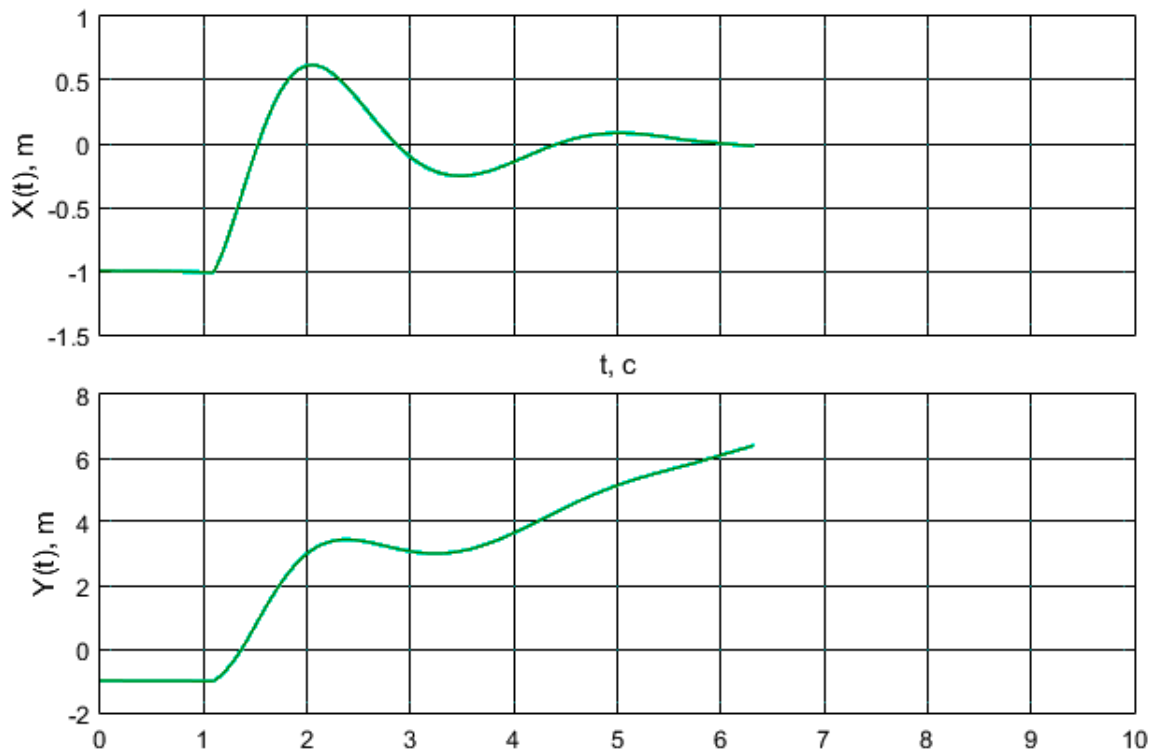


Рис. 16. Графики изменения горизонтальных координат квадрокоптера.

По графику видно, что после старта квадрокоптер некоторое время сохранял свою горизонтальную позицию. В это время происходил набор высоты до появления цели в поле зрения камеры.

Цель впервые была обнаружена на камере на высоте около 5 метров. После посадки центр масс квадрокоптера оказался смещен на 6-7 см от центра платформы, что является следствием неспособности камеры



захватывать цель полностью во время финальной стадии снижения.

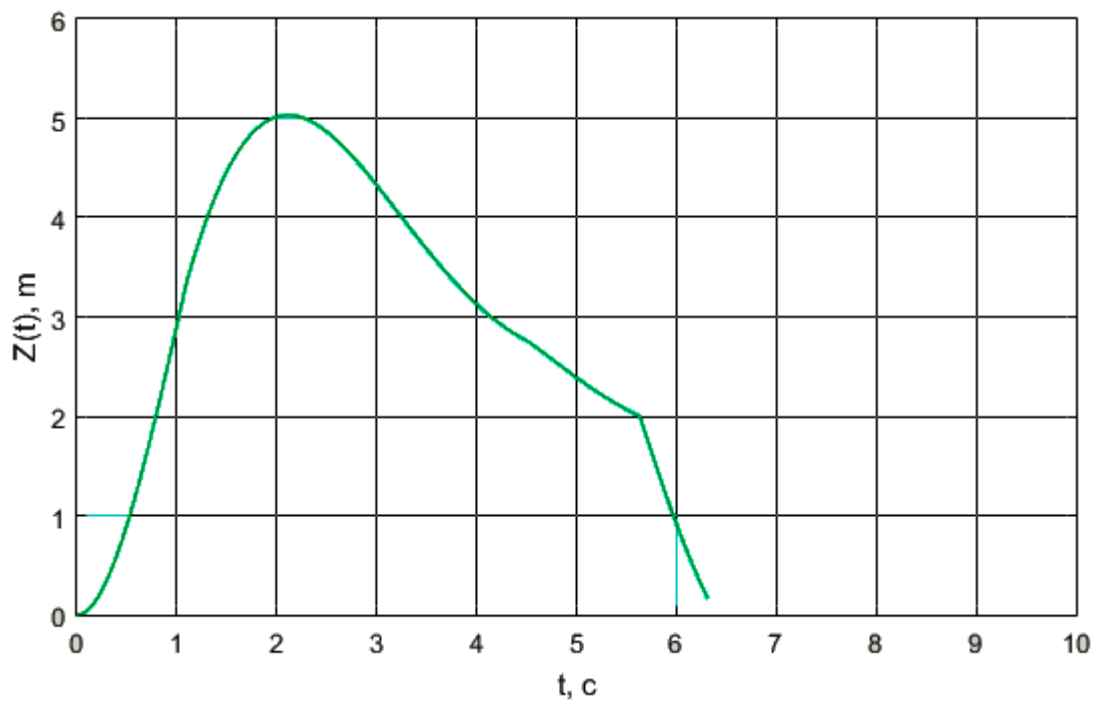


Рис. 17. График изменения высоты полета квадрокоптера.

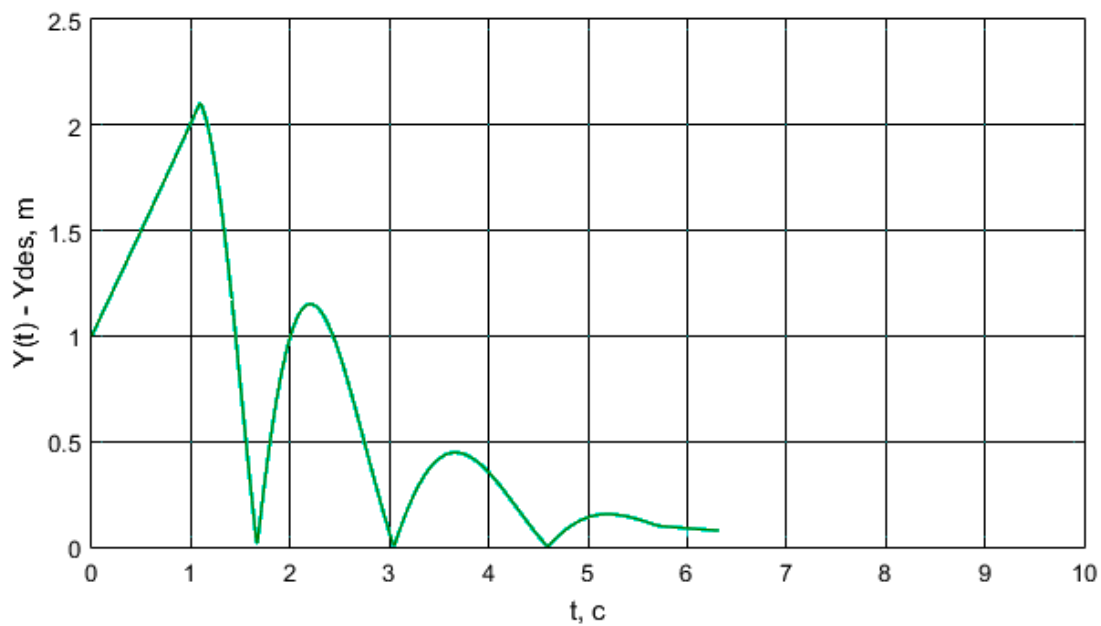


Рис. 18. График отклонения движения от прямолинейного.

## Выводы

В данной работе была рассмотрена проблема автоматической посадки квадрокоптера, основанной на методах компьютерного зрения.

В ходе работы построена нелинейная математическая модель динамики квадрокоптера, представленная в среде имитационного моделирования Simulink, а также программно реализован аналог бортовой камеры.

Для удовлетворения требования максимально точной посадки квадрокоптера синтезирован регулятор движения Backstepping, обеспечивающий следование желаемой траектории. Данный метод основан на теории Ляпунова, следовательно, результатом его является асимптотическая сходимость траектории движения к желаемой. Подход Backstepping является одним из самых популярных в области управления движением квадрокоптера, для которого построена нелинейная модель. В то же время, в дальнейшем необходимо построить и другие типы регуляторов (PID, Sliding и др.) с целью анализа качества/скорости работы и выбора наиболее подходящего.

В работе представлен результат решения в случае прямолинейного движения платформы и при данных предположениях продемонстрирована эффективная работа алгоритма.

При выполнении данной работы получены следующие результаты.

1. Построена математическая, имитационная модели квадрокоптера.
2. Для этой модели синтезирован закон управления Backstepping.
3. Программно смоделирована бортовая камера и разработан алгоритм обработки поступающих кадров.
4. Реализован алгоритм автоматической посадки на цель, находящуюся в поле зрения камеры.
5. Проведено имитационное моделирование полета полета квадрокоптера.

## Список литературы

- [1]. Mendes, A. S. “Vision-based automatic landing of a quadrotor UAV on a floating platform”, Master of Science thesis, Faculty of Aerospace Engineering, Delft University of Technology, 2012
- [2]. G.M. Hoffmann, H. Huang, S.L. Waslander, C.J. Tomlin, “Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment”, American Institute of Aeronautics and Astronautics, pp. 01-20, 2007.
- [3]. Y.M. Younes, M.A.A. Jarrah and A.A. Jhemi, “Linear vs Nonlinear Control Technology for a Quadrotor Vehicle”, Proceeding of the 7th International Symposium on Mechatronics and its Application, Shajah, UAE, April 20-22, 2010.
- [4]. T. Madani and A. Benallegue, “Backstepping Control for a Quadrotor Helicopter”, Proceeding of International Conference on Intelligent Robots and System, pp. 3255-3260, October 9-15, 2006.
- [5]. L. R. G. Carrillo, E. Randon, A. Sanchez, A. Dzul and R. Lozano, “Stabilization and trajectory tracking of a quad-rotor using vision”, Journal of Intelligent and Robotic Systems, 61(1-4), 2011, pp 103- 118.
- [6]. Sinopoli, B., Micheli, M., Donato, G., and Koo, T. J., “Vision based Navigation for an Unmanned Aerial Vehicle,” In Proceedings of IEEE International Conference on Robotics and Automation, 2001, pp. 1757–1765.
- [7]. S. Bouabdallah, A. Noth and R. Siegwart, “PID vs LQ Control Techniques applied to an Indoor Micro Quadrotor”, International Conference on Intelligent Robots and Systems, vol.3, pp. 2451 – 2456, 2004.
- [8]. S. Bouabdallah P. Murrieri and R. Siegwart, “Design and control of an Indoor Micro Quadrotor”, Proceeding of IEEE International Conference on Robotics and Automation New Orleans, pp. 4393-4398, April 2004.
- [9]. S. Bouabdallah and R. Siegwart, “Backstepping and Sliding mode Techniques Applied to an Indoor Micro Quadrotor”, Proceeding of IEEE International Conference on Robotics and Automation Barcelona, Spain, pp. 2259-2264, April 2005.