Занятие 3

Задача автоматического следования по траектории.

1. Waypoints

Наша задача по управлению лодкой на верхнем уровне сводится к тому, что нам необходимо динамически строить траекторию движения нашей лодки. Обычно, задача управления по траектории решается с использованием путевых точек (англ. Waypoint). Идея путевых точек умозрительная и понятна – это точки в пространстве, которые задают траекторию движения. Подобно тому, как вы задаете по точкам график какой-либо функции. Остальная часть траектории между точками получается путем экстраполяции. Waypoint могут задаваться вручную пользователем или программистом, а могут строиться динамически, программно. По Waypoint-ам до сих пор двигаются боты в компьютерной игре, автомобили компьютерных соперников в гоночных играх и др. А также, множество задач навигации роботов решается именно вокруг этого понятия.

В нашем случае, мы можем каким-то образом описать траекторию движения лодки, заставив ее обходить маяки. Т. е. waypoint-ы мы задаем сами. Тогда нам остается решить всего одну задачу – заставить наши роботизированные лодки двигаться от одного waypoint-а до другого.

2. Подходы к решению задачи управления.

Эта задача решается с применением различны регуляторов.

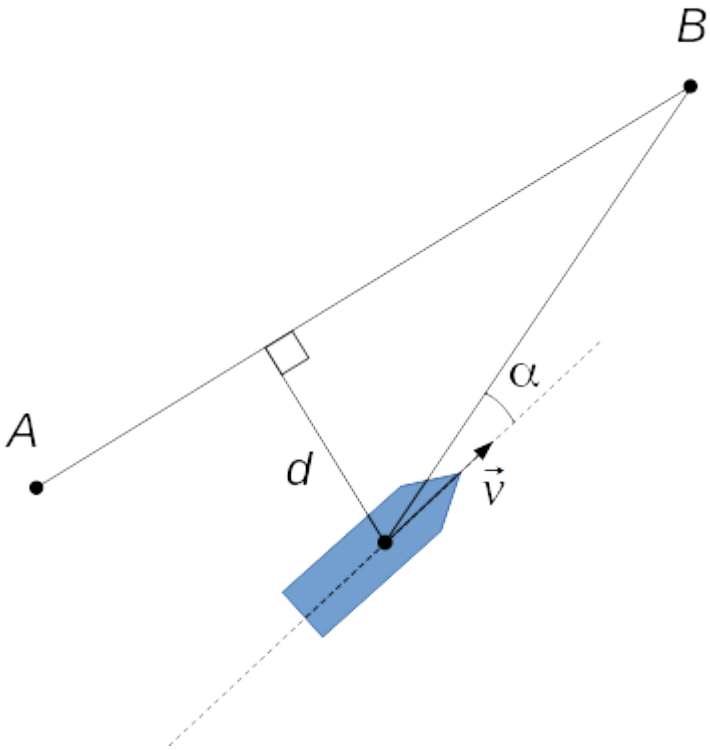
Расширяя границы познания, скажем, что существует два подхода к решению этой задачи. Первый подход, объединяющий в себе относительно простые, широко используемые алгоритмы, основанные на разделении этой задачи на два цикла управления: внешнего цикла (outer loop) и внутреннего цикла (inner loop). Внутренний цикл отвечает непосредственно за управления исполнительными органами объекта навигации (в нашем случае – лодки) для осуществления необходимого маневрирования – управление вектором скорости, ускорения, управления двигателями. Источником команд для внутреннего цикла служит внешний цикл, в котором обрабатывается стратегия движения, решаются геометрические и кинематические задачи навигации и движения.

Второй подход, появившийся с ростом производительности бортовых вычислительных систем, связан с объединением этих двух циклов. В алгоритмах этой группы сложно выделить два уровня. К алгоритмам этой группы относятся более сложные адаптивные алгоритмы, основанные на нечеткой логике, нейронных сетях и пр.

Мы с вами будем решать нашу задачу «по старинке». Но немного коснёмся передовых знаний в области «простых» алгоритмов .

3. Геометрическая формулировка задачи

Итак, давайте рассмотрим задачу геометрически. Для упрощения, мы изобразим задачу на плоскости, а уже позже, при решении будем учитывать, что наша планета не плоскость.



Итак, у нас есть точка *A* – исходная точка, из которой мы начинаем движение. И точка *B* – точка, в которую нам необходимо прибыть. Обе эти точки – это соседние waypoint-ы. Очевидно, что оптимальная траектория, без отсутствия каких-либо препятствия на пути следования (мы принимаем что препятствий нет) – это отрезок соединяющий эти две точки. И в первом приближении, наша задача реализовать такой алгоритм движения лодки, чтобы траектория ее движения как можно больше совпадала с этой линией.

Но в мире ничего не бывает идеального, поэтому поместим нашу лодку чуть в стороне от этой прямой и рассмотрим получившийся геометрический рисунок.

Вектор скорости лодки обозначим традиционного – $\vec{v}$. Отрезок *d*, перпендикулярный отрезку *AB* – это кратчайшее расстояние от текущей координаты лодки до прямой оптимальной траектории. Это расстояние в литературе называют *cross-track error*, или в переводе - *боковое отклонение от курса*. Уже из этого рисунка становится видно, дальнейшая постановка задачи зависит от того, как мы выберем целевую функцию.

В самом простом варианте, мы можем задать, что решением нашей задачи на каждом цикле управления будет минимизация угла пеленга (в англ. терминологии — *bearing*, угла между текущим курсом (*heading*) лодки и отрезком соединяющим текущую координату лодки с целевой координатой – точкой *B*). Такая стратегия называется *Line-of-sight (LOS) guidance* – следование по линии видимости. Т. е. в каждой точке маршрута, где бы мы не оказались, задача алгоритма – минимизация угла пеленга. Угол пеленга — это результат, выходное воздействие всего нашего алгоритма управления. Если он равен 0 — это означает, что наша лодка идет курсом точно в точку *B*. Это, кстати, один из самых распространенных принципов наведения различных систем вооружения.

Однако, рассмотренный алгоритм наведения (минимизация угла пеленга) имеет одну значимую слабую сторону.

Задание. Подумайте и скажите, в чем заключается слабость алгоритма? Подсказка: как он будет работать в условиях статического внешнего возмущения, например — бокового течения или ветра.

Мы специально употребляем множество иностранных терминов. На сегодняшний день наука и инженерия глобальны и множество полезных материалов существует только на самом международном языке — английском. Важно, чтобы вы умели использовать эти материалы.

Можно предложить и другие целевые функции. Например, мы можем разработать более комплексный алгоритм, который минимизирует как угол, так и *боковое отклонение от курса d*.

4. Подготовка к решению.

Для начала давайте ограничимся самым простым вариантом – где бы мы не находились, будем минимизировать угол пеленга, направляя лодку на цель. Для решения задачи управления, нам потребуется вспомнить те полезные знания математики и географии, которые, сидя на уроке в классе, вы не представляли где применять.

Вспомним, что наша планета вообще говоря не плоскость, что бы там не говорили последователи теории плоской Земли. Вы знаете, что она даже не сфера, а эллипсоид, очень близкий к сфере. И вот эта близость нам очень поможет в расчетах, а именно позволит их упростить. Вычисление расстояние и азимута между двумя точками на Земле, заданными двумя геодезическими координатами (именно эти координаты дает нам приемник GPS: долгота и широта, выраженные в градусах) — классическая геодезическая задача.

Очень хороший материал по способам вычисления расстояния приведен здесь:

http://gis-lab.info/qa/great-circles.html

В том материале не содержится теоретических сведений о расчете азимута, однако алгоритм расчета приведен в примерах листингах исходного кода.

Как считать угол между курсом и пеленгом (т. е. угол, между дугами, образованными тремя точками на сфере) приведено в следующей статье на том же ресурсе:

http://gis-lab.info/qa/angles-sphere.html

Дабы не утешить вашего интереса к решению задачи, мы не приводим здесь готовых функций, реализующих все эти расчеты. Тем не менее, подсмотреть на примеры реализации непосредственно в Arduino можно вот в этой открытой библиотеке:

https://github.com/mikalhart/TinyGPS

Некоторые особо хитрые могут применять ее как есть, не написав ни строчки кода этих функций. Но будьте осторожны — бездумное использование сторонних библиотек таит множество опасностей.

Но погодите, все это время мы говорили только о данных GPS. А для чего нам нужен компас?

Задайте себе вопрос, а можно ли вообще обойтись без компаса и использовать только GPS для определения курса? Ответ: теоретически — можно. Давайте рассмотрим почему и как

можно и почему теоретически. Как мы можем узнать азимут имея только данные от системы глобального позиционирования? Очень просто. У нас есть координата лодки на текущем шаге цикла обработки и координата лодки на предыдущем шаге цикла. Современные недорогие GPS-приемники, подобные тому, что используется вами, передают навигационную информацию в микропроцессорную систему с частотой до 10Гц. Т. е. 10 раз в секунду, наша система управления получает информацию о новых координатах объекта навигации. Имея две соседние координаты, текущую и предыдущую, можно построить вектор направления движения. А если продифференцировать координату по времени (интервал времени у нас задан жестко, 100мс), то вычислить еще и скорость лодки.

Но тут нам на пятки наступает реальность. А именно в том моменте, что точность GPS относительно невелика. И для столь малых и относительно медленно движущихся объектов она не позволяет достоверно определять ни курс, ни скорость. Предположим, скорость лодки 1 м/с. А точность GPS при самом удачном стечении спутниковой обстановки составляет ±0,5 метра. Т. е. соседние измерения координат, принятые на интервале 1/10 секунды (те самые 10Гц) окажутся внутри нашей погрешности — лодка за 100 мс пройдет путь всего 10 см. Конечно, мы можем использовать в расчете данные не с двух соседних точек, а, скажем, с каждой десятой или двадцатой. Но тогда очевидно теряем в качестве управления — система получает актуальные данные достаточно редко.

В реальности, при решении задачи управления нужно грамотно использовать оба датчика, принимая во внимание все их ограничения и возможности.

Частым очевидным решением является то, что применяется, например в популярном полетном контроллере Ardupilot (APM). Разработчики этого контроллера применяют компас для определения (корректировки) азимута при скорости объекта меньше либо равной пороговому значению 3м/с, и GPS в случае, когда объект управления движется быстрее этой скорости.

Итак, мы умеем вычислять расстояния, курс, пеленг. Нам осталось только реализовать сам регулятор. И этому будет посвящен следующий урок. А еще через урок мы опустимся на уровень ниже и рассмотрим каким образом вообще принимать и обрабатывать данные с компаса и приемника GPS.