УДК 004.896:621.865.8

*С.А. Голь, В.С. Леушкин*

# АЛГОРИТМИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПЛАНИРОВАНИЯ ЛОКАЛЬНОЙ ТРАЕКТОРИИ АВТОНОМНОГО РОБОТА НА КАРТЕ ПРОХОДИМОСТИ В РЕАЛЬНОМ ВРЕМЕНИ

*Рассматриваются недостатки алгоритмов планирования траектории и поиска пути в дискретном пространстве состояний для задачи управления автономным транспортным средством. Предложена модификация алгоритма планирования траектории автомобиля-робота в реальном времени, позволяющего планировать манёвры разворота и огибания препятствий. Рассмотрены особенности реализации и результаты практического применения предложенного алгоритма.*

***Ключевые слова**: планирование траектории в реальном времени, автомобиль-робот, автономное управление.*

**Введение.** Область применения автономных транспортных средств и их функциональные возможности постоянно расширяются. Возможными сценариями использования автомобилей-роботов являются доставка грузов, автоматическое сопровождение объектов, следование за лидером в колонне, охрана периметра, инспекция удалённых объектов, разведка местности и спасательно-поисковые работы при чрезвычайных ситуациях.

Во время выполнения сценария роботу предстоит осуществлять движение в слабоструктурированной среде с недостаточными знаниями о рельефе местности и возможных препятствиях. Движение по пересечённой местности также подразумевает появление дополнительных преград на пути следования, заранее не отмеченных на карте. Таким образом, важным требованием к программному обеспечению (ПО) автономного робота является способность реагирования на изменение дорожной обстановки и построения безопасной траектории движения в реальном времени. Особую сложность представляет планирование сложных манёвров, таких как параллельная парковка или разворот в условиях ограниченного пространства, когда габариты транспортного средства не позволяют совершить U-образный разворот. При этом появляется необходимость планирования последовательности из нескольких движений вперёд-назад, что значительно увеличивает вычислительные затраты.

Ежегодные испытания автомобилей-роботов проводятся в России и Европе для обмена опытом между разработчиками и исследователями и предоставляют возможность испытать роботов и алгоритмы в среде, имитирующей реальные условия применения.

**Рисунок 1 – Автомобиль-робот «ГАЗель» и ОРТК**

Команда СКБ РГРТУ приняла участие во всероссийских полевых испытаниях автомобилей-роботов «Робокросс-2014» и европейских испытаниях «ELROB 2014» с двумя робототехническими платформами: автомобиль-робот «ГАЗель» и отладочный робототехнический комплекс (ОРТК) (рисунок 1).

Испытания «Робокросс-2014» проводятся на специально оборудованном полигоне «Берёзовая пойма» завода «ГАЗ» с целью замены водителя при выполнении циклических заездов в будущем. Основное задание испытаний «Трек» пред-

ставляло собой сценарий автономного движения по ограниченному участку автомобильного трека, на котором расположены статические препятствия. В конце трека находились тупик и зона разворота, достигнув которую, робот должен был развернуться и вернуться в зону старта.

На испытаниях «ELROB 2014» были представлены различные сценарии использования автомобилей-роботов в полевых условиях. Команда СКБ принимала участие в задании «Мул», в котором необходимо было осуществлять автономную доставку груза между двумя пунктами по пересечённой местности, огибая такие препятствия, как окопы, заграждения из колючей проволоки и непроходимой растительности.

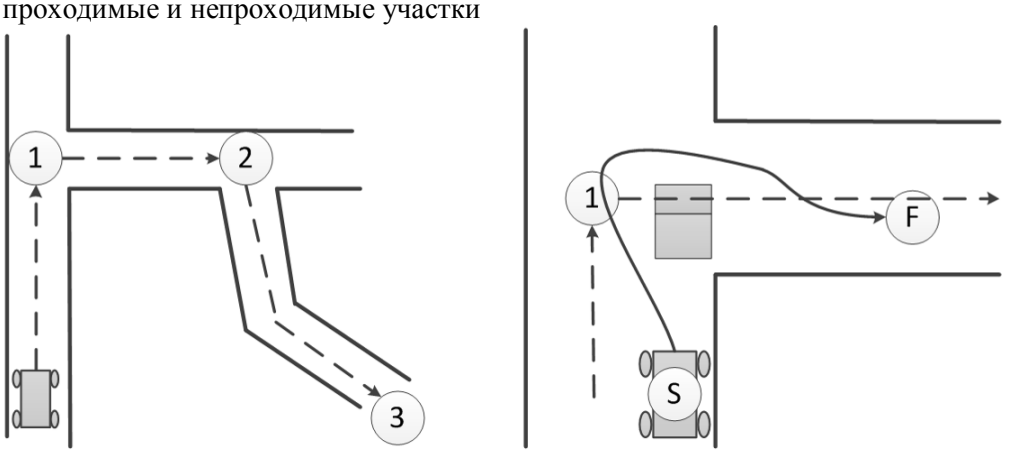
Роботы команды СКБ РГРТУ оснащены датчиками положения (GNSS приёмник Геос-3М с темпом выдачи информации о положении 10 Гц) и ориентации в пространстве (UM6-LT от CH Robotics), а также линейными лазерными сканерами SICK LMS511 (робот «ГАЗель») и 3D лазерным сканером Velodyne HDL-32E (ОРТК).

На основе данных с сенсоров осуществляется построение локальной карты местности, отражающей проходимые и непроходимые участки вокруг транспортного средства. Учитывая то, что в задании текущего сезона соревнований появилась необходимость движения задним ходом, на «ГАЗель» был установлен дополнительный лазерный сканер, осуществляющий обзор в задней полуплоскости автомобиля.

Описание общей архитектуры программного обеспечения, осуществляющего обработку сенсорных данных и принятие управляющего решения, представлено в [1].

Следует различать понятия глобальной и локальной траектории.

Планирование глобальной траектории (рисунок 2, а) представляет собой процесс выбора контрольных точек маршрута без учёта текущей дорожной обстановки. Список контрольных точек может быть известен заранее (на основе истории записей GPS координат, как в задании «Мул») либо построен на основе векторной карты дорожной сети. Такой маршрут может быть проложен до начала движения, однако для безопасного следования необходимо корректировать маршрут, реагируя на изменение дорожной обстановки в реальном времени.




**а** **б**

**Рисунок 2 – Маршрут и траектория транспортного средства:**
**а – маршрут следования автомобиля (пунктир);**
**б – локальная траектория (сплошная) и промежуточная позиция F**

Планирование локальной траектории – получение траектории, которая учитывает расположение дороги и препятствий, а также кинематические и динамические ограничения автомобиля и положение ключевых точек маршрута [1]. Оно осуществляется в ограниченном радиусе вокруг автомобиля (порядка 20-80 метров), который определяется размерами используемой карты проходимости (рисунок 2, б).

На основе результата планирования и текущего положения робота управляющее ПО осуществляет выбор скорости и положения руля автомобиля для следования по выбранной траектории.

Таким образом, планирование локальной траектории направлено на исключение столкновений с препятствиями и выхода автомобиля с определённого дорогой коридора движения.

**Целью данной работы** является модификация алгоритма планирования локальной траектории CL-RRT для управления автомобилем-роботом в реальном времени.

**Описание проблемы.** Планирование локальной траектории для автомобиля-робота – одна из ключевых задач при управлении транспортным средством.

Разработанные для дискретного пространства состояний алгоритмы поиска кратчайшего

пути, такие как A* [2], неприменимы для непрерывного пространства состояний. Также не учитываются [1] кинематические ограничения движения автомобиля. Модификации алгоритма, такие как Hybrid-A* [3], учитывают указанные особенности, однако результирующая траектория требует дополнительного сглаживания, так как получается неоправданно извилистой.

Алгоритмы, использующие метод опорных векторов [4], позволяют сразу получить сглаженную траекторию, но, как и Hybrid A*, требуют дополнительной модификации для генерации сложных манёвров, таких как разворот с применением заднего хода и парковка.

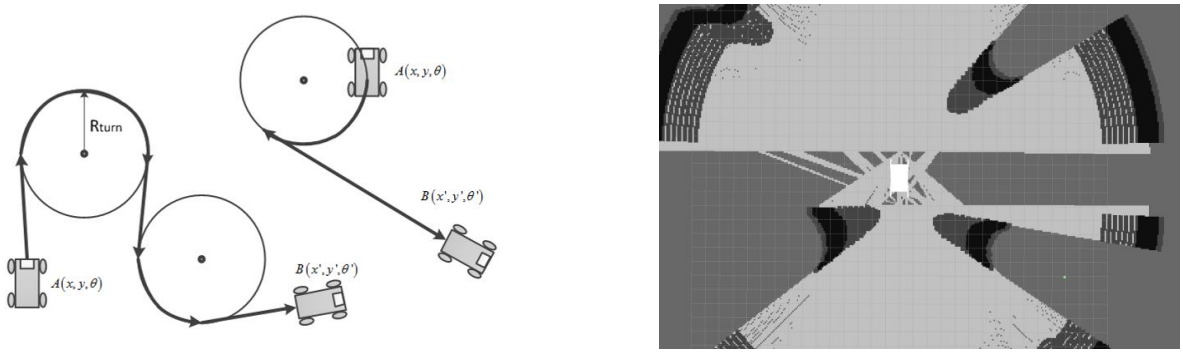
В общем случае процесс поиска кратчайшего пути в непрерывном пространстве состояний представляет собой неразрешимую задачу, так как перебор всех состояний невозможен. Именно поэтому были разработаны методы, позволяющие получить траекторию, близкую к оптимальной. Эти методы основаны, как правило, на случайном поиске. Известным представителем данной группы методов является RRT – Rapidly exploring Random Tree – быстро разворачивающееся случайное дерево [5].

Однако применение RRT напрямую невозможно ввиду того, что этот метод не учитывает кинематические ограничения робота-автомобиля.

Kuwata и другими в работе [6] представлена модификация метода RRT – CL-RRT, позволяющая учитывать указанные выше ограничения.

В работе [1] был представлен алгоритм планирования траектории, основанный на CL-RRT, который был с успехом применён на испытаниях автомобилей-роботов «Робокросс-2013». Однако ввиду отсутствия возможности планирования разворота задним ходом и недостаточной производительности возникла необходимость переработки алгоритма.

**Описание алгоритма.** Для описания траектории в настоящей работе используется путь Ридса – Шеппа (Reeds – Shepp path) [7] – способ представления траектории в виде набора дуг и отрезков прямых. Данный способ позволяет описать траекторию движения автомобиля из положения $A(x,y,\theta)$ в положение $B(x',y',\theta')$ с учётом ограничения минимального радиуса поворота автомобиля $R_{turn\_min}$ (рисунок 3, а). Сложные траектории, такие как манёвр объезда препятствия, строятся из набора элементов пути Ридса – Шеппа. Манёвр движения задним ходом описывается аналогично, однако скорость движения на данном участке считается отрицательной. Отличие от пути Дьюбинса (Dubins path) [8], использованного в предыдущей работе, заключается в том, что движение задним ходом по пути Дьюбинса невозможно.




**а** **б**

**Рисунок 3 – Пример возможных траекторий:**
**а – пример путей Ридса – Шеппа;**
**б – карта проходимости, полученная с помощью дальномеров SICK, закреплённых впереди и сзади автомобиля**

Робот оснащён дальномерами для получения информации об окружающих его препятствиях. В процессе обработки данных строится карта проходимости (рисунок 3,б), где белым помечены проходимые участки, а чёрным – непроходимые. Серые оттенки характеризуют проходимую, но нежелательную для движения область и область вблизи препятствий. Все это позволяет осуществлять планирование траектории на безопасной удалённости от препятствий и двигаться с более высокой скоростью. Однако в случае невозможности построения траектории полностью в безопасной зоне траектория будет проложена непосредственно вблизи препятствий, при этом допустимая скорость движения может быть снижена для безопасного перемещения.

Задачу планирования локальной траектории определим следующим образом.

Даны точки текущего положения автомобиля $S(x,y,\theta)$ и цели $F(x',y',\theta')$, карта проходимости с функцией стоимости $M(x,y,\theta)$, где $M \in [0;100]$, $M < 100$ означает проходимый уча-

сток. Необходимо построить такую траекторию Ридса – Шеппа $D$, которая переводит автомобиль из состояния $S$ в состояние $F$, чтобы каждая точка траектории $p(x,y,\theta)$ являлась проходимой.

Метод RRT заключается в итеративном росте дерева возможных траекторий так, чтобы участки траектории, соединяющие смежные вершины, были проходимы. За начальную вершину роста дерева принимается исходная позиция. Позиции промежуточных вершин выбираются случайно на каждой итерации.

Рост дерева прекращается при достижении целевой позиции, что свидетельствует об успешно проложенной траектории. Если в процессе роста дерева превышено допустимое количество итераций, то считается, что построить траекторию не удалось. В этом случае рост дерева прекращается и робот останавливается до тех пор, пока корректная траектория не будет построена.

Алгоритм CL-RRT схож с RRT, главное отличие заключается в способе проверки проходимости между соседними вершинами. В RRT считается, что робот голономен и размерами робота можно пренебречь. Участок траектории, соединяющий две вершины $S_1$ и $S_2$, считается проходимым, если каждая точка на отрезке между вершинами является проходимой.

Общая схема предложенного алгоритма планирования локальной траектории представлена ниже.

**Алгоритм «Рост дерева»**

*Инициализация*

1. Корень дерева = начальная позиция.
2. Сформировать множество из $T_{Nmax}$ промежуточных положений $T$.
3. Добавить в $T$ целевые положения.
4. Список нерассмотренных вершин = все вершины дерева.

*Итерации.*

*Для каждой нерассмотренной вершины дерева:*

5. Если (Уровень узла < $L_{max}$) :
   1) выполнить процедуру «Рост вершины»;
   2) добавить новые вершины в список нерассмотренных.

*Для полученных возможных траекторий:*

6. Сортировать «возможные траектории» по длине.
7. Результат – траектория с наименьшим индексом дальности.

**Алгоритм процедуры «Рост вершины»**

*Для каждого $i$-го положения из множества $T$:*

1. Рассчитать индекс дальности $l$ от вершины до позиции $T_i$.
2. Добавить во временный список пару $\langle l,i \rangle$.
3. Сортировать список по возрастанию $l$.
4. Выбрать не более $V_{max}$ элементов из списка таких, что путь из текущей вершины в точку $T_i$ является проходимым.
5. Если среди выбранных элементов есть целевая точка, то пометить текущую и все предшествующие вершины как «возможная траектория».
6. Присоединить выбранные элементы к текущей вершине дерева.

Результатом работы алгоритма «Рост дерева» является дерево возможных траекторий. Дерево решения представляет собой связанный набор вершин. Каждая вершина характеризуется координатами $(x,y,\theta)$, а также кривой Ридса – Шеппа, по которой необходимо проследовать, чтобы попасть в них. Также вершина имеет флаг «возможное решение», который характеризует принадлежность вершины к траектории, которая приводит в целевую точку.

На этапе инициализации корнем дерева является начальное положение робота. Затем осуществляется процесс выборки промежуточных вершин.

Ввиду того, что данный алгоритм предназначен для планирования траектории автомобиля, процесс выбора промежуточных состояний осуществляется из множества положений, лежащих не на всей карте проходимости, а лишь в некоторой области впереди автомобиля (как в [6]). Это позволяет сократить объём вычислений и заранее отсечь нерациональные траектории.

Выборка промежуточных позиций на шаге 2 основана на проверке проходимости в случайной точке $S^R(s_x,s_y,\theta)$ с координатами, рассчитанными по формуле[6]:

$$s_x = x_0 + r\cos(\theta),$$
$$s_y = y_0 + r\sin(\theta),$$

где $r = \sigma_r |n_r| + r_0$, $\theta = \sigma_\theta n_\theta + \theta_0$, $n_r$ и $n_\theta$ – скалярные случайные величины, распределённые по нормальному закону $n_r \sim N(0,1)$ и $n_\theta \sim N(0,1)$ соответственно, $\sigma_r$ и $\sigma_\theta$ – коэффициенты, характеризующие разброс точек в продольном и поперечном направлении относительно положения автомобиля $(x_0,y_0,\theta_0)$ соответственно, $r_0$ и $\theta_0$ характеризуют центр области разброса точек в полярных координатах относительно центра робота по дальности и азимуту соответственно.

В результате выбирается $T_{Nmax}$ промежуточных точек $T$, к которым добавляются целевые точки (в частном случае – одна).

Для реализации возможности планирования заднего хода на втором шаге алгоритма к множеству промежуточных положений $T$ добавляются позиции, помеченные флагом «движение назад». При обработке данных положений на следующих этапах алгоритма разница заключается лишь в учёте отрицательной скорости движения транспортного средства.

Следует отметить, что постоянное добавление вершин с флагом «движение назад» существенно уменьшает производительность и может приводить к генерации нерациональных траекторий, поэтому осуществляется только в определённых условиях. При невозможности построить траекторию в течение некоторого количества итераций принимается решение о необходимости добавления особых положений. Генерация позиций для движения задним ходом осуществляется не только впереди транспортного средства, но и вокруг него по тем же формулам, что и для движения вперёд, только $n_r$ берётся без знака модуля, а $r_0$ считается равным нулю:

$$r = \sigma_r n_r .$$

Метод добавления «специальных» вершин особенно удобен не только для решения задачи планирования последовательности действий, ограниченных кинематикой робота, но и, наоборот, для расширения допустимых движений. Так, для ОРТК была реализована возможность разворота на месте без дополнительных изменений алгоритма путём добавления специальной позиции «разворот».

На шаге 5 осуществляется этап «роста дерева», на котором итеративно строится дерево траекторий, покрывающих карту проходимости, при этом для каждой вершины дерева, уровень узла которой меньше порога $L_{max}$, из списка нерассмотренных вершин вызывается алгоритм «Рост вершины».

Алгоритм «Рост вершины» осуществляет поиск ближайших по индексу дальности вершин из множества $T$ к текущей вершине.

Индекс дальности вычисляется как сумма значений проходимости $M(x,y,\theta)$ для всех точек, принадлежащих траектории. Если одно из положений является непроходимым, то вся траектория считается недопустимой.

Затем первые $V_{max}$ допустимых положений в порядке возрастания индекса дальности добавляются к дереву и списку нерассмотренных вершин.

Выбор наименьших расстояний при соединении соседних вершин позволяет получить сглаженную траекторию и избавиться от нерациональных и потенциально опасных траекторий [6].

Рост дерева прекращается, когда невозможно добавить новые вершины либо превышено количество итераций.

На шаге 6 осуществляется поиск траектории с наименьшим индексом дальности среди узлов, помеченных флагом «возможная траектория». Результирующая траектория представляет собой набор кривых Ридса – Шеппа, следуя по которым, транспортное средство будет перемещаться от начальной вершины к целевой (рисунок 4).

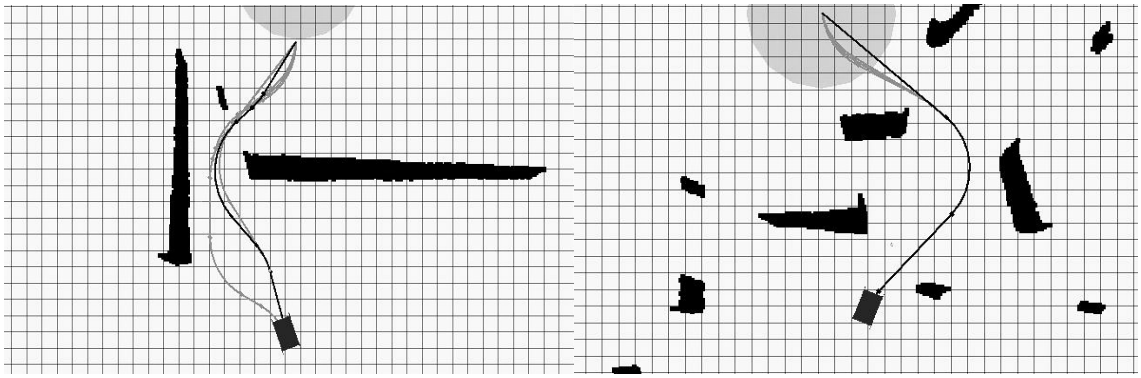


**Рисунок 4 – Пример полученных траекторий (светлые линии) и лучшая траектория (темная линия)**

Так как алгоритм осуществляет перестройку траектории на каждой итерации, то вследствие случайного выбора промежуточных вершин кратчайшая траектория может не совпадать с проложенной ранее, а также вовсе отсутствовать (ввиду малых $L_{max}$ или $T_{Nmax}$ и сложной дорожной ситуации). Как было отмечено в [6] и на практике, это приводит к «волнистому» движению робота, что является нежелательным, так как может привести к неустойчивому управлению и непредсказуемости поведения для остальных участников движения. Следует отметить, что постоянная перестройка траектории является вычислительно трудоёмкой.

Решением данной проблемы является не полная перестройка, а обновление дерева решений. В процессе движения вдоль траектории проверяется корректность выбранного решения. Если меняется дорожная обстановка, то прежние узлы траектории до участка пересечения с препятствиями сохраняются и служат «точкой роста» на этапе инициализации алгоритма.

Эвристика обновления дерева была изменена ввиду недостаточной производительности для движения на максимальной регламентированной скорости в течение испытаний.

После каждой итерации роста дерева осуществляется удаление тех веток траекторий, которые длительное время не были помечены как «возможная траектория». Это даёт возможность поддерживать дерево в актуальном состоянии и значительно сократить количество вычислительных затрат на следующей итерации, когда осуществляется проверка на проходимость существующего множества траекторий.

Из множества возможных решений выбирается $D_{max}$ кратчайших траекторий, в то время как остальные отсекаются. Оставшиеся $D_{max}$ траекторий дополнительно разбиваются на несколько элементов, тем самым создавая возможность «роста» дерева из заведомо удачных позиций на следующей итерации.

Данные изменения позволили увеличить одновременно число «возможных траекторий» и, как следствие, общую производительность алгоритма.

Полученная траектория передаётся на исполнение подпрограмме управления движением, которая использует контроллер «упреждающего преследования», как в [6].

Подпрограмма планирования локальной траектории была реализована на языке C++ и выполнялась на компьютере с процессором Core-i5 3.0 ГГц. Частота работы алгоритма составила 10 Гц, что достаточно ввиду ограничения скорости правилами соревнований (10 км/ч), и в 2 раза быстрее, нежели в предыдущей работе [1].

В ходе испытаний «Робокросс-2014» ОРТК в пробных заездах развивал скорость до 15 км/ч.

Пример карты проходимости и полученной траектории показан на рисунке 4.

**Заключение**. Представленная модификация алгоритма CL-RRT позволяет осуществлять планирование траектории автомобиля-робота в реальном времени с учётом неголономности транспортного средства. Алгоритм реализует планирование объезда препятствий и следования в контрольную точку маршрута, выполнения U-образного разворота и разворота задним ходом в условиях ограниченного пространства. Команда СКБ РГРТУ стала абсолютным победителем во всех номинациях испытаний беспилотных роботов «РобоКросс-2014» .

Результаты проведённых испытаний позволяют судить о потенциальных возможностях алгоритма для расширения набора выполняемых манёвров. Предполагается реализовать учёт динамики подвижных препятствий при планировании траектории, а также учёт дорожной обстановки при движении в колонне транспортных средств.

***Библиографический список***

1. *Жулев В.И., Леушкин В.С., Нгуен Т.Н.* Планирование локальной траектории автомобиля-робота в реальном времени // Вестник Рязанского государственного радиотехнического университета. 2013. № 4-3 (46). С.18-23.

2. *Hart P. E., Nilsson N. J., Raphael B.* A Formal Basis for the Heuristic Determination of Minimum Cost Paths // IEEE Transactions on Systems Science and Cybernetics SSC4 4 (2): 100–107, 1968.

3. *Dolgov D., Thrun S., Montemerlo M., Diebel J.* Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments // I. J. Robotic Res., 29, 485-501., 2010.

4. *Qingyang C., Zhenping S., Daxue L., Fang Yuqiang F., Xiaohui L.* Local Path Planning for an Unmanned Ground Vehicle Based on SVM // International Journal of Advanced Robotic Systems, 2012.

5. *LaValle S. M.* Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University, October 1998.

6. *Kuwata Y. et al.* Real-time Motion Planning with Applications to Autonomous Urban Driving, 2009.

7. *Reeds J. A., Shepp L. A.* Optimal paths for a car that goes both forwards and backwards // Pacific Journal of Mathematics, 145 (2): 367–393, 1990.

8. *Dubins L.E.* On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents // American Journal of Mathematics 79 (3): 497–516, July 1957.