Введение в робототехнику

Лекция 12.

Часть 2. Навигация и картографирование.

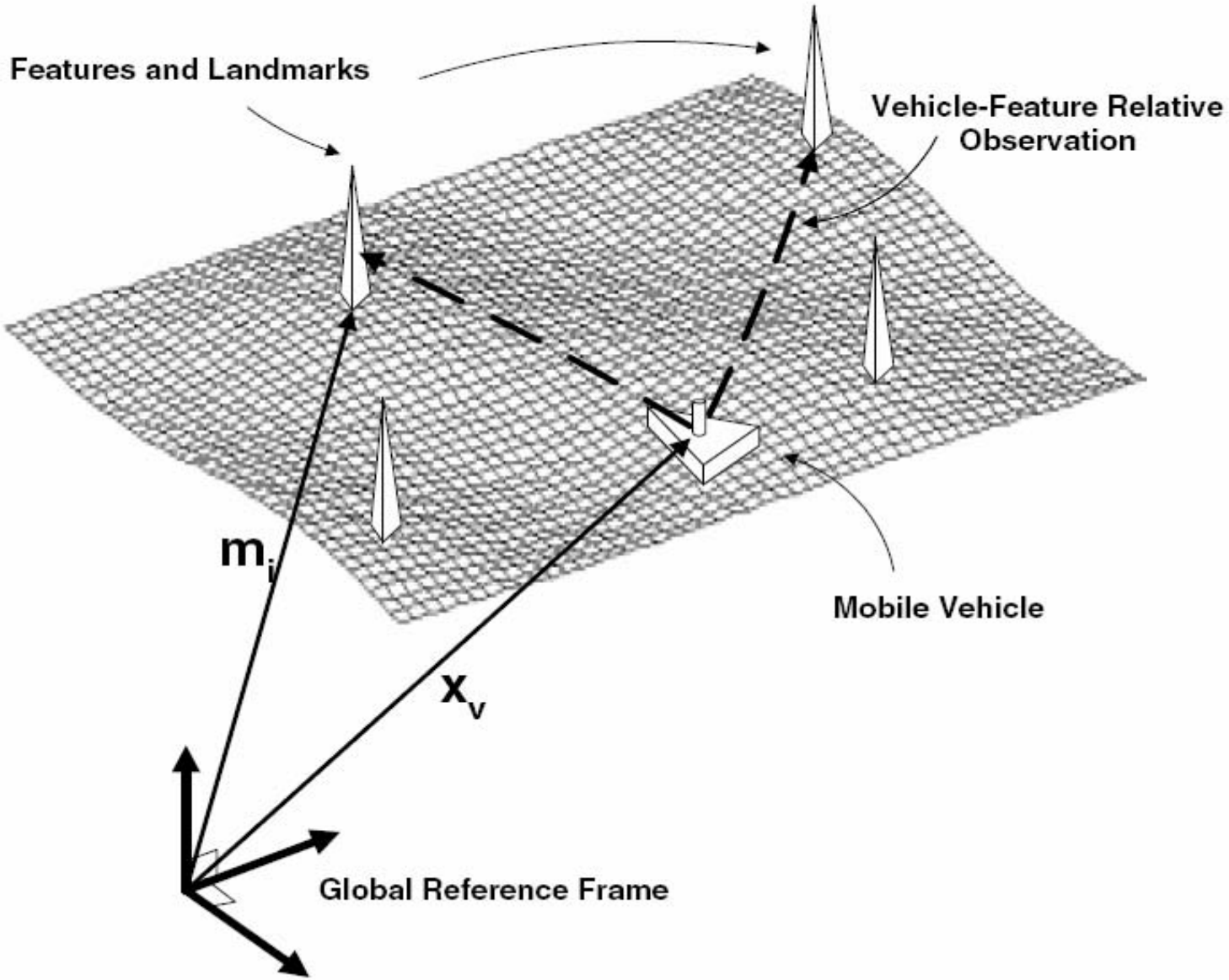
SLAM

SLAM

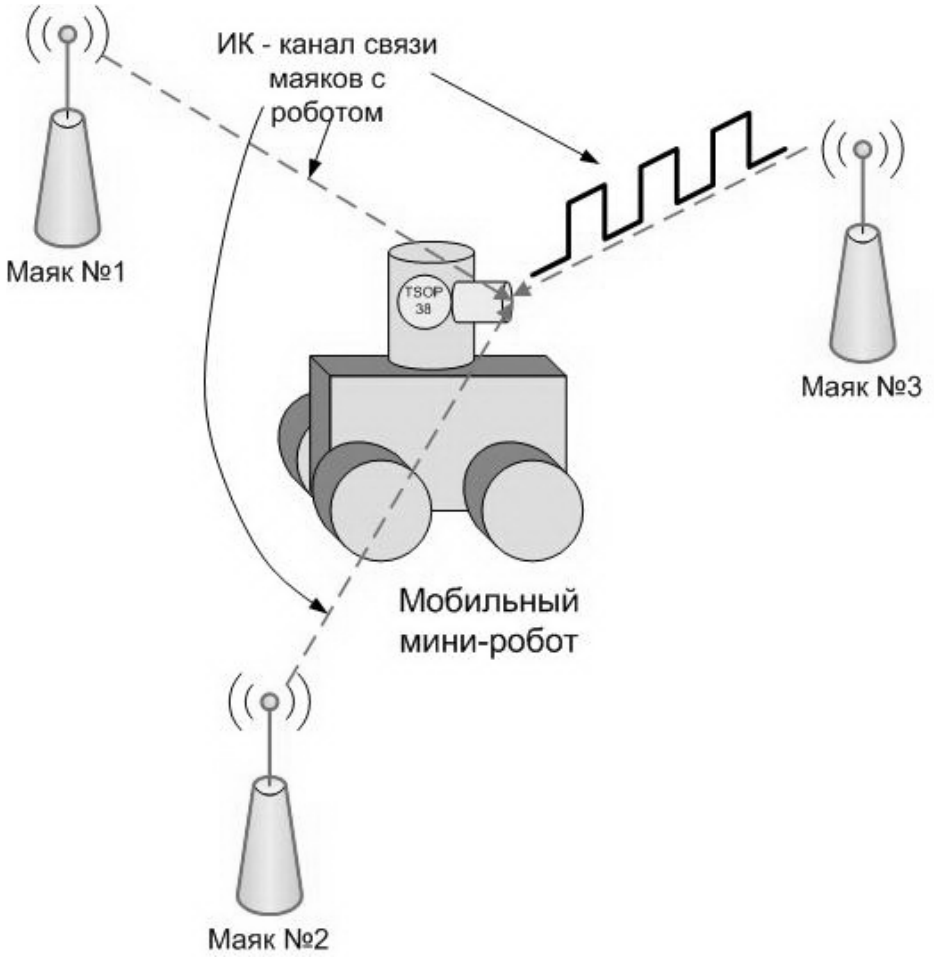
- Simultaneous Localization And Mapping (одновременная локализация и картографирование)
- Задача SLAM является одной из наиболее изведанных областей робототехники (изучается примерно с 1990 года) и заключается в одновременном выполнении процессов картографирования роботом окружающей местности и идентификации себя на построенной карте.

Пример с дистанционно управляемым мобильным роботом

- Колеса связанные с двигателем и камеры, а так же приборы для контроля скорости и направления
- Оператор принимает решение в каком направлении двигаться роботу и при помощи камер может оценить окружающую обстановку и построить карту
- В этом примере человек выполняет работу, описанную в задаче SLAM.


Features and Landmarks
Vehicle-Feature Relative Observation
m_i
Mobile Vehicle
x_v
Global Reference Frame

Мобильный робот в окружении маяков – источников инфракрасного излучения

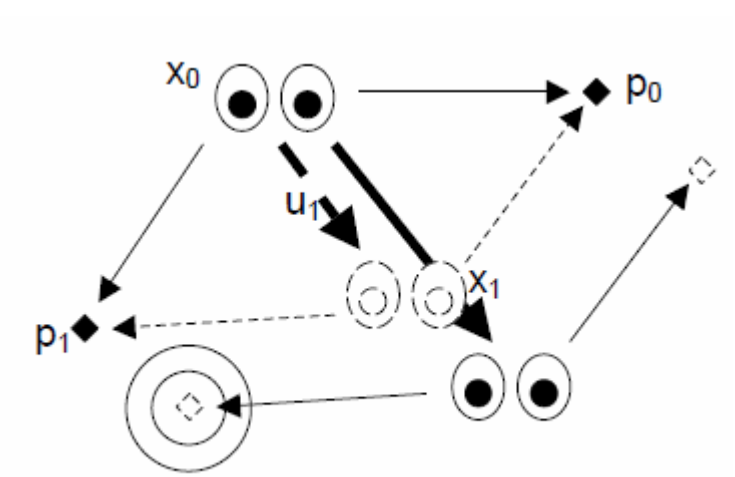


Локализация

- Предположим, что мобильный робот находится в местности содержащей маяки - точки в пространстве с известным местоположением, благодаря которым робот может вычислить свое расположение, зная расстояние до них
- Предположим, что робот находится в некотором известном месте X0, и знает о некотором множестве P содержащим несколько других локаций.
- Когда робот попытается двигаться в заданном направлении на заданное расстояние (вектор U1) в точку X1, то на самом деле он будет двигаться относительно некоторого другого вектора в некоторое место находящееся вблизи с X1.
- Теперь информация о маяках должна быть переопределена с учетом нового положения робота. Если принять что данные полученные с приводов робота абсолютно точны то информацию о маяках можно использовать повторно сместив их положение относительно вектора U1. Но так как информация полученная с приводов не является абсолютно точной то робот должен заново получить информацию об окружающих маяках. Поиск начинается от маяков находящихся поблизости текущей локации к маякам находящимся на удалении

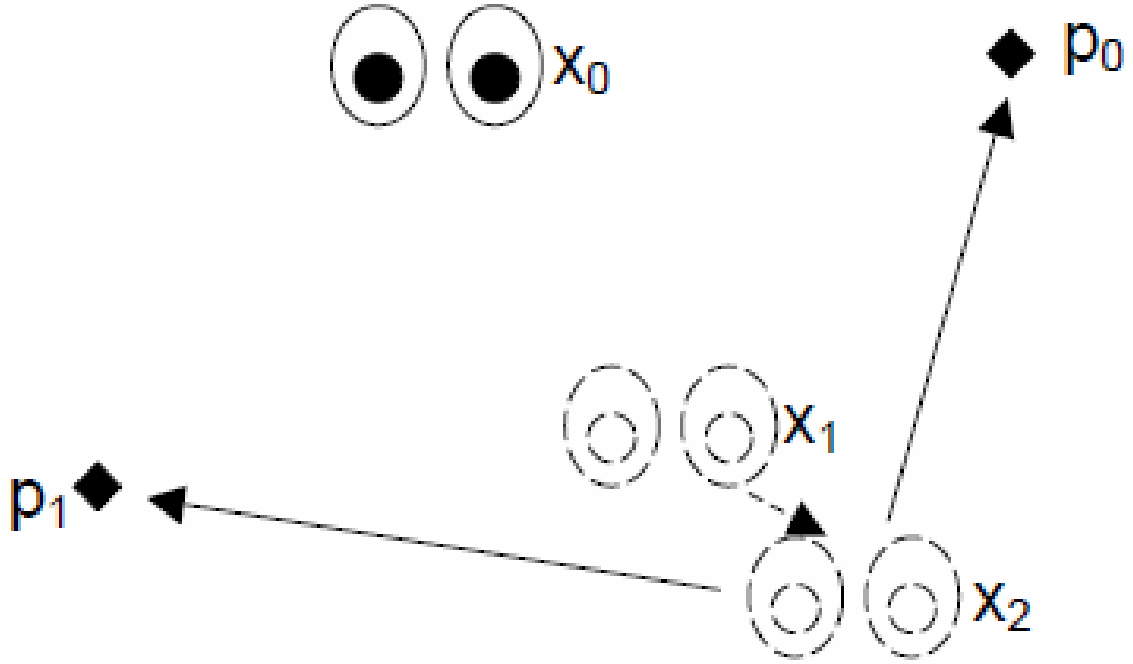
Локализация (2)

P0, P1 - маяки



точка X1 соответствует оценочному вектору перемещения,
а не вектору соответствующему реальному перемещению.
Для следующего перемещения в точку X2 маяки используется повторно,
как показано на далее:

Локализация (3)



С течением времени оценка положения робота не отличается от настоящего положения, поскольку расположение маяков является абсолютным и новое положение робота рассчитывается по правильным данным.

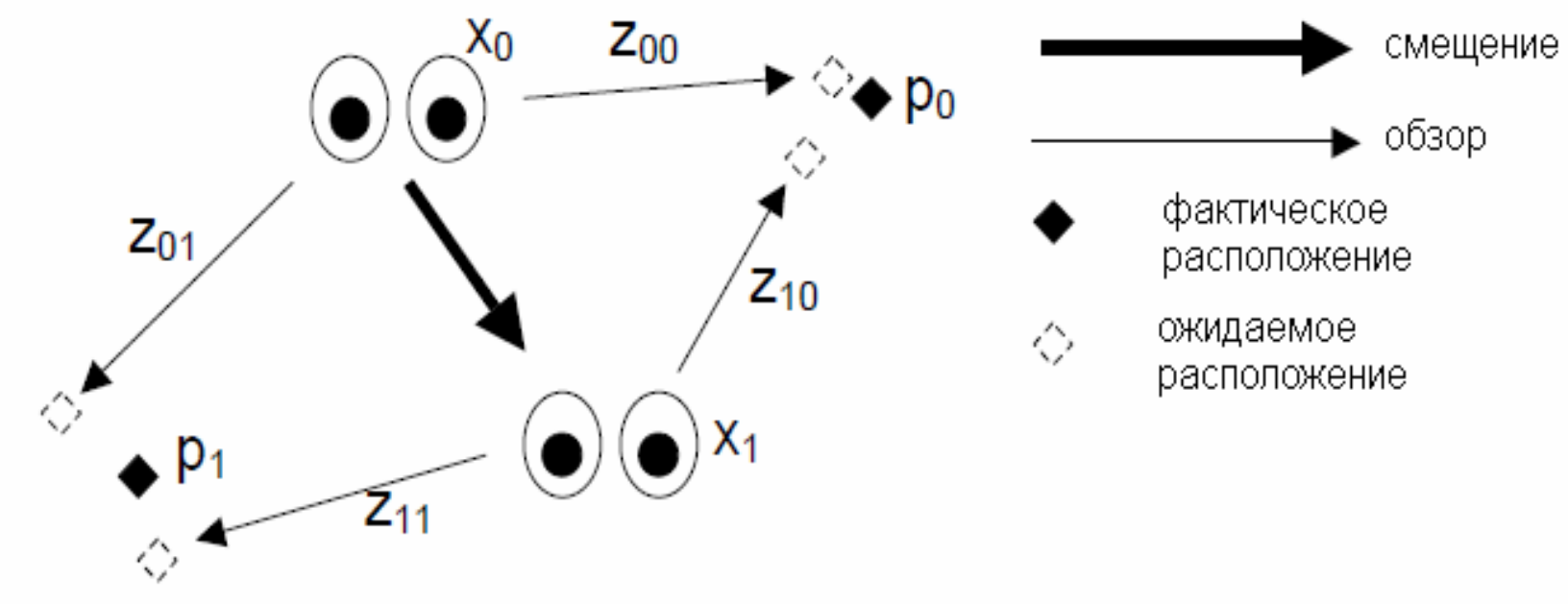
Картографирование

- В картографии предполагается, что роботу известно где он находится в любой момент времени, то есть, мы либо имеем абсолютно точные датчики движения, или безошибочный GPS
- В отличие от локализации робот не имеет точной информации об окружающем мире
- Есть возможность построить карту $Z_0 = \{Z_{00}, Z_{01}, ... Z_{0n}\}$, где Z_{ij} можно обозначить как оценка j-й точки на временном промежутке i. Конечно, Z_0 будет содержать приближение фактического расположения некоторой точки. Однако со временем при перемещении робота карта может становиться более точной

Картографирование (2)

- Рассмотрим ситуацию из предыдущего примера, когда робот двигался из точки X1 в точку X2 вдоль вектора U1. Для начала роботу опять необходимо узнать некоторый набор реальных точек, но на этот раз особенные точки будут восприниматься сдвинутыми относительно своего настоящего положения с учетом ошибок полученных с датчиков, а не из-за одометрических ошибок. Ситуация похожа на предыдущую, однако могут быть использованы точки находящиеся вблизи с роботом для облегчения их поиска. Как только это будет сделано, можно будет построить еще одну карту Z1 содержащую новые точки с учетом карты Z0

Картографирование (3)



Картографирование (4)

- Теперь можно создать карту Z2, которая будет сочетать информацию из Z0 и Z1, но будет содержать только один экземпляр каждой точки. Одна из идей заключается в минимизации общего расстояния между двумя восприятиями места в каждой j-й точке.
- То есть z2j - элемент карты z2 - будет рассчитываться из соответствующих элементов z0j и z1j карт Z0 и Z1. В нашем двумерном примере эта задача заключается в выборе точки на прямой, соединяющей точки z0j и z1j, которые являются разными восприятиями одной точки i. Таким образом, новые оценки положений точек станут более точными. Этого метода вполне достаточна исключительно в контексте картографии, хотя он не совсем оптимален.
- Другой вариант минимизации заключается в использовании квадрата расстояния, а не общего расстояния. Это решение является более перспективным и будет рассмотрено подробней далее.

Картографирование (5)

- Вне зависимости от выбора метода можно предположить что точность карты Zi будет улучшаться с увеличением i. Пока наш робот движется и получает новые данные с датчиков, оценки точек на карте Zi будут становиться все более точными. Но это при условии правильной калибровки датчиков. Если какой-то датчик не был откалиброван то оценка положений не сможет улучшаться

SLAM

- Каждый из вышеописанных алгоритмов требует каких-то входных данных, которые, как правило, не доступны в реальной ситуации
- Задача картографирование заключается в проблеме сопоставления информации полученной с датчиков робота. Она может быть описана вопросом: "Как выглядит окружающий мир?“. Основная проблема картографирования заключается в получении представления об окружающей среде на основе данных полученных с датчиков робота.
- С другой стороны необходимо решить проблему локализации, то есть получить оценку положения робота на построенной им же карте местности. Иными словами робот должен уметь отвечать на вопрос: "Где я?" Как правило, отсчитывают разницу между начальным положением робота и текущим
- SLAM решает одновременно две проблемы: построение карты местности окружающей робота и в то же время определение положения робота на этой карте. На практике эти две проблемы не могут быть решены независимо друг от друга
- Поэтому SLAM часто называю т проблемой "курицы и яйца": для локализации необходима подробная карта и в то же время для построения карты необходима точная оценка положения робота

Алгоритм SLAM

1. Робот находится в некотором неизвестном месте. Он начинает строить карту видимой местности со своей позиции
2. Выбирает новое место для дальнейшего передвижения
3. Передвигается на новое место и сравнивает свое текущее положение с ожидаемым, полученным на предыдущем шаге
4. Обновляет карту, используя предыдущие и новые данные и переходит к шагу 2.

В SLAM используются:

- Фильтр Калмана
- Метод Монте-Карло

Фильтр Калмана

- Это эффективный рекурсивный фильтр, который оценивает состояние линейной динамической системы по серии неточных измерений
- Используется в широком спектре задач от радаров до систем технического зрения, и является важной частью теории управления системами
- Разработан еще в 1960 году. Однако не получил широкого распространения до 1988 года
- Фильтр Калмана является разновидностью рекурсивного фильтра. Это означает, что только результат предыдущей итерации фильтра (в виде оценки состояния системы и оценки погрешности определения этого состояния) и текущие наблюдения нужны для расчета текущего состояния системы

Фильтр Калмана (2)

- Данный фильтр обрабатывает входные данные и возвращает их оценочные значения.
- В контексте SLAM возвращаемые переменные это положение робота и особых точек местности относительно окружающего мира. Входные данные могут содержать данные с датчиков привода, датчиков движения и т.п.
- Фильтр Калмана использует известную нам математическую модель динамики объекта, которая описывает какие вообще изменения состояния объекта возможны, чтобы устранить погрешности измерения и предоставить хорошей точности положение объекта в данный момент (фильтрация), в будущие моменты (предсказание), или в какие-то из прошедших моментов (интерполяция или сглаживание).

Фильтр Калмана (3)

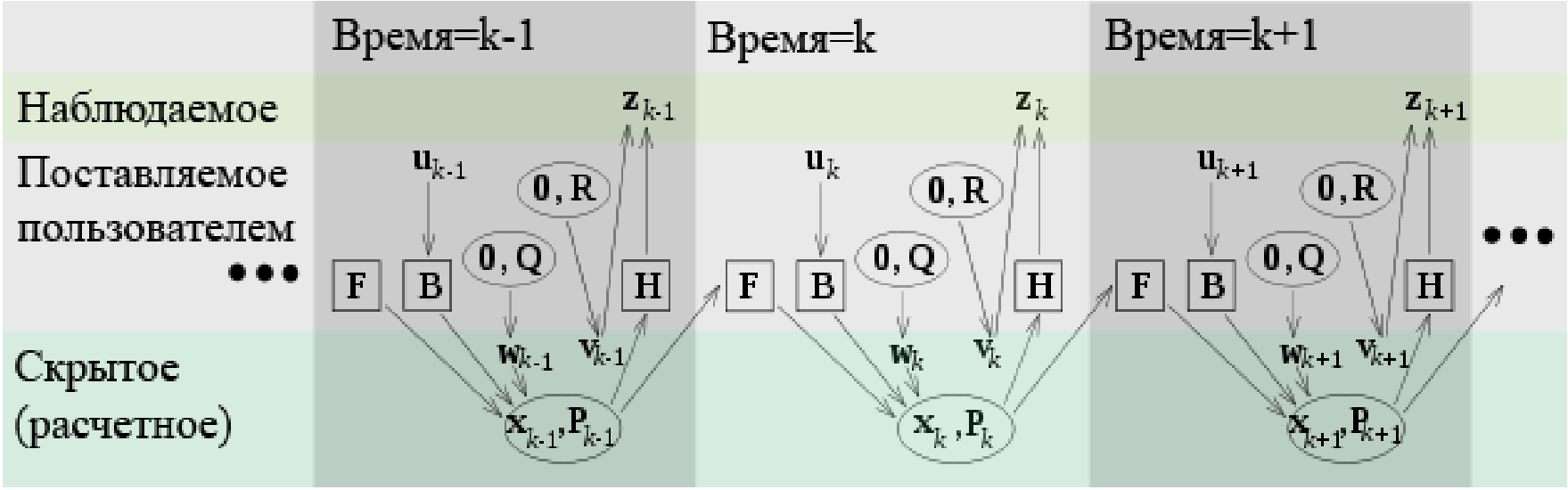
- Фильтры Калмана основываются на линейных динамических системах, дискретизированных по времени. Они моделируются цепями Маркова, построенными на линейных операторах с внесенными погрешностями с нормальным Гауссовым распределением.
- Состояние системы считается вектор из действительных чисел
- При каждом шаге по времени, линейный оператор применяется к вектору состояния системы, добавляется некоторая погрешность и опционально некоторая информация об управляющих воздействиях на систему, если таковая известна
- После чего другим линейным оператором с другой погрешностью добавляется видимая информация о состоянии системы

Фильтр Калмана (4)

- Фильтр Калмана можно рассматривать в качестве аналога скрытым моделям Маркова, с тем ключевым отличием, что переменные, описывающие состояние системы, являются элементами бесконечного множества действительных чисел (в отличие от конечного множества пространства состояний в скрытых моделях Маркова). Кроме того, скрытые модели Маркова могут работать с произвольными распределениями для следующих значений переменных состояния системы, в отличие от модели стандартного Гауссового распределения, поддерживаемого фильтрами Калмана. Существует строгая взаимосвязь между уравнениями фильтра Калмана и аналогичными в скрытых моделях Маркова

Фильтр Калмана (5)

- Чтобы было возможным использовать фильтр Калмана для оценки внутреннего состояния системы по серии неточных измерений, необходимо представить модель данного процесса в соответствии с универсальной моделью процесс для фильтра Калмана. Это означает, что нужно указать матрицы **F***k*, **H***k*, **Q***k*, **R***k*, и иногда **B***k* для каждого шага по времени *k*, как указано ниже
- Квадратиками обозначены матрицы. Эллипсами обозначены нормальные распределения (с указанными в скобках матрицами матожиданий и ковариантностей). Значения без кружочка и квадратика вокруг являются векторами



Фильтр Калмана (6)

- Модель системы для фильтра Калмана подразумевает, что реальное состояние в момент времени *k* получается из состояния в момент времени (*k* − 1) по правилу:

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k$$

,

- где
- **F***k* матрица, соответствующая модели преобразованию системы со временем, применяемая к предыдущему состоянию **x***k*−1;
- **B***k* матрица соответствующая модели применения управляющего воздействия, которая применяется к состоянию системы умноженная на вектор управляющего воздействия **u***k*;
- **w***k* вектор погрешности, которая как предполагается, имеет нулевое матожидание, нормальное Гауссово распределение и матрицу ковариаций **Q***k*

Фильтр Калмана (7)

- В момент времени *k* производится наблюдение (или измерение) **z***k* реального состояния системы **x***k* в соответствии с моделью измерения

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$$

где

- **H***k* матрица, соответствующая модели наблюдения, которая отображает пространство векторов реального состояния системы в пространство векторов результатов наблюдений,
- **v***k* это вектор ошибки наблюдения, который считается имеющим нулевое матожидание, нормальное Гауссово распределение и матрицу ковариаций **R***k*

Метод Монте-Карло (для локализации)

- Метод локализации Монте-Карло (или MCL - Monte Carlo Localization) основанный на фильтре частиц - это относительно новый подход к решению проблемы локализации робота
- Данный метод использует коллекцию образов так же известную как “частицы“. Каждая частица характеризуется как возможное положение робота в данный момент времени, ей соответствует вероятность того что робот находится именно на этой частице. Строго говоря, это вероятности, но их так же часто называют весами
- Чем большим числом частиц мы располагаем, тем быстрее программа будет сходиться к правильному решению, но за счет большего вычислительного времени

Метод Монте-Карло (2)

- Когда робот начинает движение он еще не знает где находится, поэтому вероятность расположения робота в какой-то частице распространяется равномерно по всем частицам. Со временем вероятность нахождения на частице, располагающейся вблизи робота, начнет увеличиваться, а на удаленных частицах – уменьшаться
-

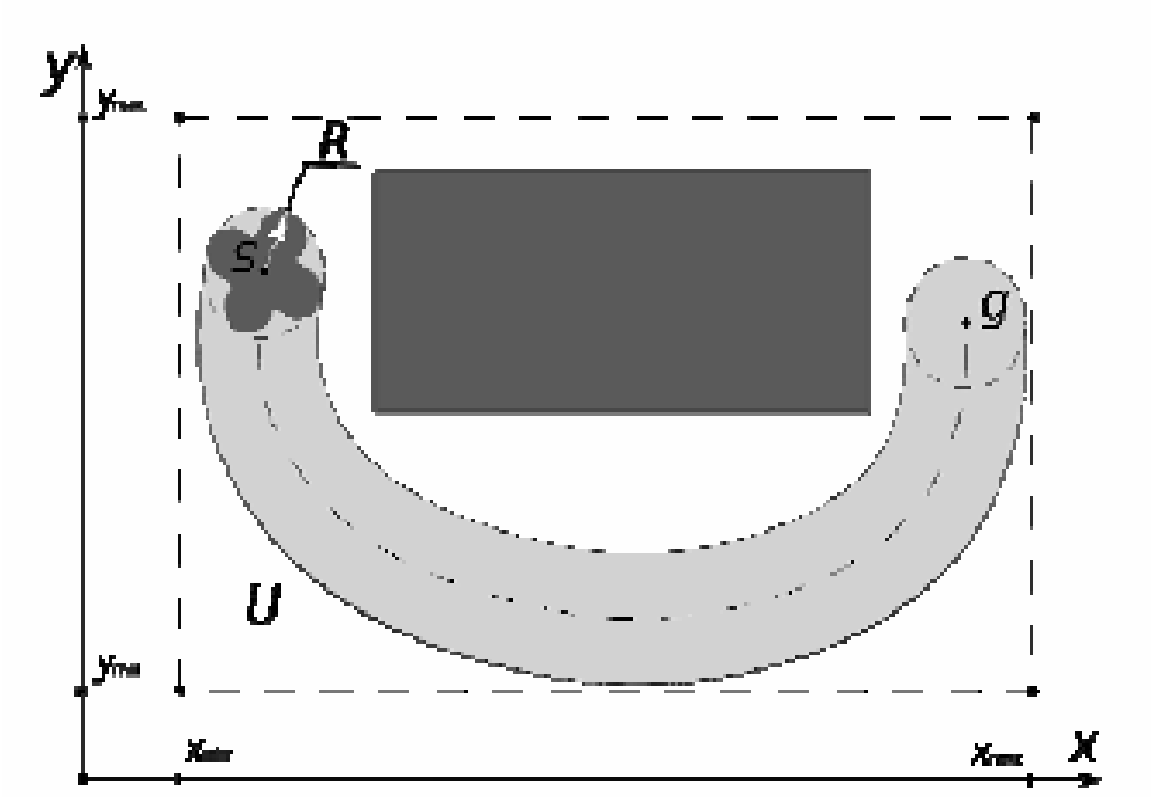
Метод Монте-Карло (3) Алгоритм:

- Инициализируется набор частиц таким образом, что их расположение относительно друг друга равномерно распределено, и их веса равны.
- Повторять для текущего набора частиц:
  - переместить робота на фиксированное расстояние а затем снять показания датчиков
  - Обновление информации о расположении каждого из образцов (основываясь на модели движения)
  - Назначение весов каждой частице, на основании датчиков для выбора нового приближенного расположения робота
  - сделать новую коллекцию частиц используя частицы старой коллекции на основании вероятностных значений весов.
  - Сделаем эту коллекцию текущей

Планирование траектории движения робота

- **В 2-мерном пространстве**
  - Наземные мобильные роботы, в т.ч. роботы-автомобили
- В 3-мерном пространстве
  - Подводные мобильные роботы
  - Воздушные мобильные роботы
  - Роботы-манипуляторы

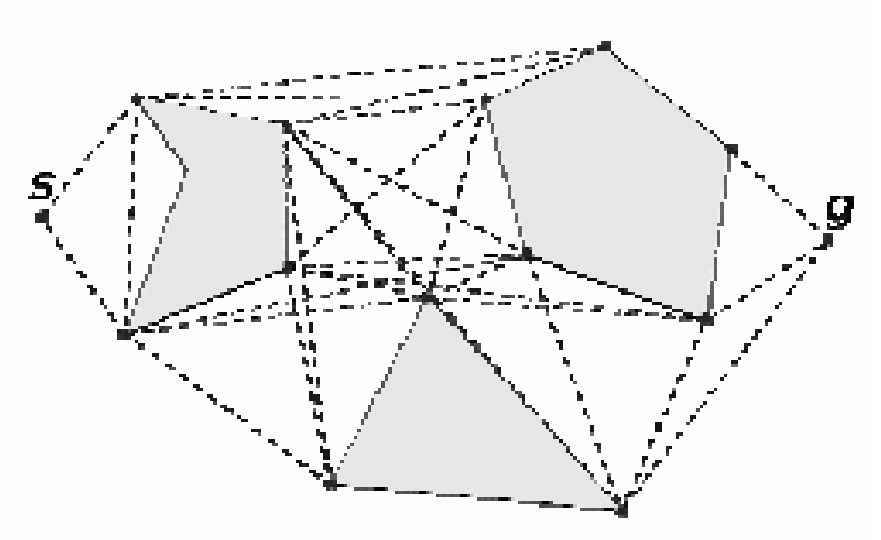
Допустимая траектория в задаче планирования



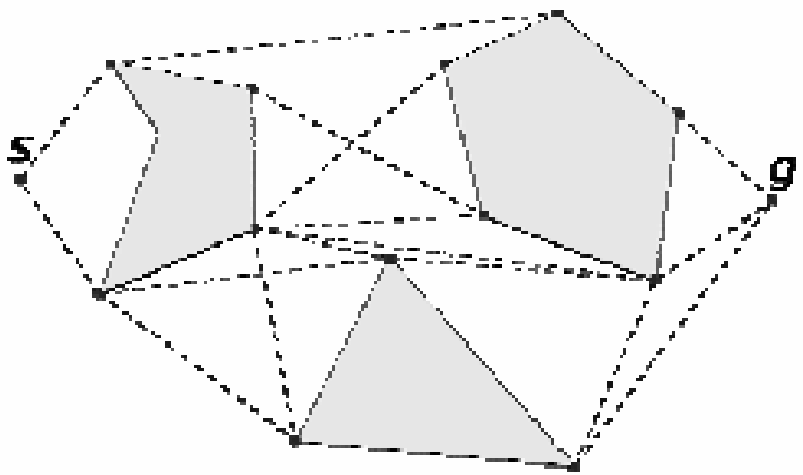
R – робот
g - цель

Представление окружающего пространства с препятствиями

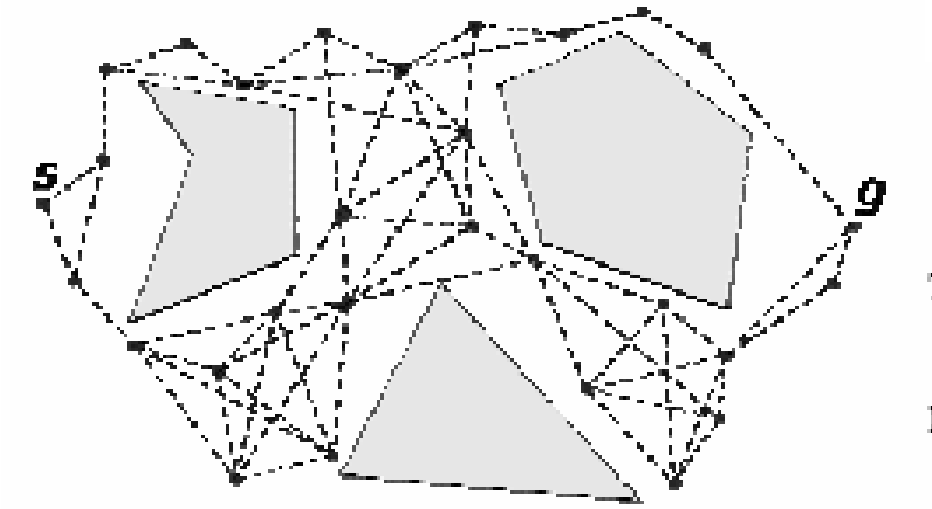
Граф видимости



Ограниченный граф видимости



Вероятностная схема местности



Алгоритм построения:

Шаг 1. Добавить точки s и g в граф.

Шаг 2. Выбрать точку из рабочей области.

Шаг 3. Если выбранная точка непроходима, то перейти к шагу 2.

Шаг 4. Добавить выбранную точку в граф, а именно:

Шаг 4.1. Для каждой точки в графе проверить проходим ли отрезок прямой, соединяющий текущую и выбранную на шаге 2 точку.

Шаг 4.2. Если отрезок проходим, то добавить соответствующее ребро в граф (с весом равным длине отрезка).

Шаг 4.3. Перейти к шагу 2.

Построения графа местности методом регулярной декомпозиции

Шаг 1. Положить $i = 0$, $j = 0$, $p_x = x_{min} + R$, $p_y = y_{max} - R$.

Шаг 2. Если круг радиуса r с центром в точке (p_x, p_y) лежит в U и является проходимым, то добавить в граф вершину, соответствующую точке (p_x, p_y), с индексными координатами (i, j).

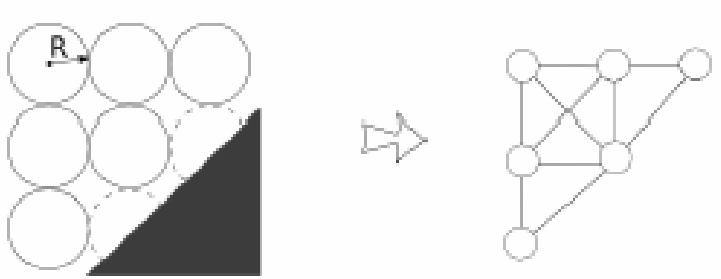
Шаг 3. Положить $j = j + 1$.

Шаг 4. Положить $p_x = p_x + 2R$.

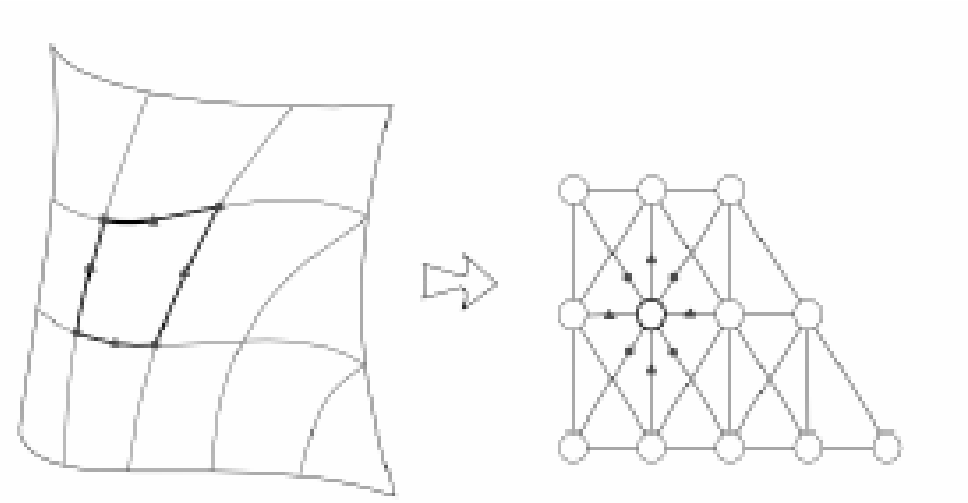
Шаг 5. Если точка (p_x, p_y) лежит в рабочей области, то перейти к шагу 2.

Шаг 6. Если $p_x > x_{max}$ и $p_y < y_{min}$, то завершить работу алгоритма.

Шаг 7. Положить $j = 0$; $i = i + 1$; $p_y = p_y - 2R$; $p_x = x_{min} + R$ и перейти к шагу 2.



Метрический топологический граф (МТ-граф)



Поверхность

МТ-граф

Клеточное представление местности