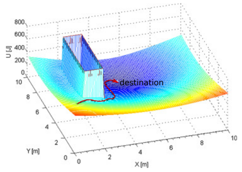
Математические задачи мобильной робототехники: навигация, автономность и управление движением при коммуникационных ограничениях

A. Matveev

Saint Petersburg state University,
Scientific and Technological University “Sirius”
almat1712@yahoo.com



Методы автономной навигации, опирающиеся на дискретизацию сцены

Общие определения и примеры

Общие определения и примеры

Sample-based representation of the scene

- The continuum of points in the free space is represented by finitely many “delegates” (samples)
- Edges signal that there is a known way to transfer the robot from the “tail” sample to the “head” one

A graph (either directed or undirected)

Общие определения и примеры

Sample-based representation of the scene

- The continuum of points in the free space is represented by finitely many “delegates” (samples)
- Edges signal that there is a known way to transfer the robot from the “tail” sample to the “head” one

A graph (either directed or undirected)

Two phases

1. Building the graph (sample-based representation)
2. Using the graph (based on graph search)

Общие определения и примеры

Sample-based representation of the scene

- The continuum of points in the free space is represented by finitely many "delegates" (samples)
- Edges signal that there is a known way to transfer the robot from the "tail" sample to the "head" one

A graph (either directed or undirected)

Two phases

1. Building the graph (sample-based representation)
2. Using the graph (based on graph search)

Single- and multi-query usage

- Single query: a single and a priori known pair "(start sample)-(destination sample)" should be served. These samples are among the nodes of the graph. The overall search process is terminated as soon as the way between these samples is found
- Multi-query: many and a priori unknown pairs should be served. 1 is run until the scene is comprehensively described. Search of the way to the graph (roadmap) and from it is needed.

Общие определения и примеры

Sample-based representation of the scene

- The continuum of points in the free space is represented by finitely many “delegates” (samples)
- Edges signal that there is a known way to transfer the robot from the “tail” sample to the “head” one

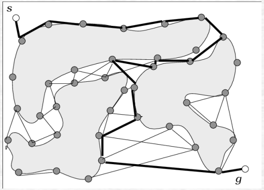
A graph (either directed or undirected)

Two phases

1. Building the graph (sample-based representation)
2. Using the graph (based on graph search)

Single- and multi-query usage

- Single query: a single and a priori known pair “(start sample)-(destination sample)” should be served. These samples are among the nodes of the graph. The overall search process is terminated as soon as the way between these samples is found
- Multi-query: many and a priori unknown pairs should be served. 1 is run until the scene is comprehensively described. Search of the way to the graph (roadmap) and from it is needed.



Общие определения и примеры

Sample-based representation of the scene

- The continuum of points in the free space is represented by finitely many “delegates” (samples)
- Edges signal that there is a known way to transfer the robot from the “tail” sample to the “head” one

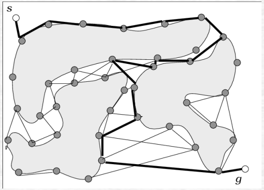
A graph (either directed or undirected)

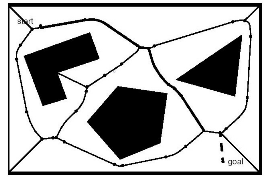
Two phases

1. Building the graph (sample-based representation)
2. Using the graph (based on graph search)

Single- and multi-query usage

- Single query: a single and a priori known pair “(start sample)-(destination sample)” should be served. These samples are among the nodes of the graph. The overall search process is terminated as soon as the way between these samples is found
- Multi-query: many and a priori unknown pairs should be served. 1 is run until the scene is comprehensively described. Search of the way to the graph (roadmap) and from it is needed.





Общие определения и примеры

Sample-based representation of the scene

- The continuum of points in the free space is represented by finitely many “delegates” (samples)
- Edges signal that there is a known way to transfer the robot from the “tail” sample to the “head” one

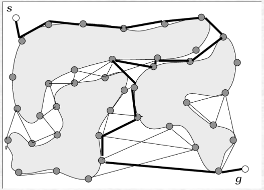
A graph (either directed or undirected)

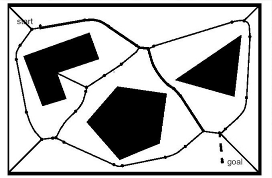
Two phases

1. Building the graph (sample-based representation)
2. Using the graph (based on graph search)

Single- and multi-query usage

- Single query: a single and a priori known pair “(start sample)-(destination sample)” should be served. These samples are among the nodes of the graph. The overall search process is terminated as soon as the way between these samples is found
- Multi-query: many and a priori unknown pairs should be served. 1 is run until the scene is comprehensively described. Search of the way to the graph (roadmap) and from it is needed.





Completeness and succession of the phases

- 1 and 2 are run in parallel or 1 is periodically paused to run 2
- 1 is run until the scene is comprehensively described

Общие определения и примеры

Sample-based representation of the scene

- The continuum of points in the free space is represented by finitely many “delegates” (samples)
- Edges signal that there is a known way to transfer the robot from the “tail” sample to the “head” one

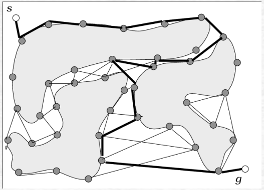
A graph (either directed or undirected)

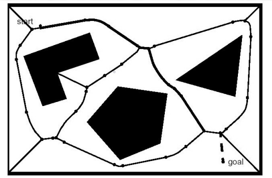
Two phases

1. Building the graph (sample-based representation)
2. Using the graph (based on graph search)

Single- and multi-query usage

- Single query: a single and a priori known pair “(start sample)-(destination sample)” should be served. These samples are among the nodes of the graph. The overall search process is terminated as soon as the way between these samples is found
- Multi-query: many and a priori unknown pairs should be served. 1 is run until the scene is comprehensively described. Search of the way to the graph (roadmap) and from it is needed.





Completeness and succession of the phases

- 1 and 2 are run in parallel or 1 is periodically paused to run 2
- 1 is run until the scene is comprehensively described

Criteria of completeness

- Deterministic connectivity of the result
- With probability 1, eventually finds a way
- Finds a way provided that the resolution of some basic underlying sampling structures is high enough

Общая схема и компоненты

Общая схема и компоненты

“First cover and then select” approach

Общая схема и компоненты

“First cover and then select” approach

- Build sampling of a large and simple domain that covers both workspace and the obstacles

Общая схема и компоненты

"First cover and then select" approach

- Build sampling of a large and simple domain that covers both workspace and the obstacles
- Discard samples that lie in obstacles

Общая схема и компоненты

“First cover and then select” approach

- Build sampling of a large and simple domain that covers both workspace and the obstacles
- Discard samples that lie in obstacles
- Link connectable samples with edges

Общая схема и компоненты

“First cover and then select” approach

- Build sampling of a large and simple domain that covers both workspace and the obstacles
- Discard samples that lie in obstacles
- Link connectable samples with edges

Constituents

Общая схема и компоненты

“First cover and then select” approach

- Build sampling of a large and simple domain that covers both workspace and the obstacles
- Discard samples that lie in obstacles
- Link connectable samples with edges

Constituents

Общая схема и компоненты

“First cover and then select” approach

- Build sampling of a large and simple domain that covers both workspace and the obstacles
- Discard samples that lie in obstacles
- Link connectable samples with edges

Constituents

1. Generator of samples

Общая схема и компоненты

“First cover and then select” approach

- Build sampling of a large and simple domain that covers both workspace and the obstacles
- Discard samples that lie in obstacles
- Link connectable samples with edges

Constituents

1. Generator of samples
2. Detector of obstacles

Общая схема и компоненты

“First cover and then select” approach

- Build sampling of a large and simple domain that covers both workspace and the obstacles
- Discard samples that lie in obstacles
- Link connectable samples with edges

Constituents

1. Generator of samples
2. Detector of obstacles
3. Local planner

Общая схема и компоненты

“First cover and then select” approach

- Build sampling of a large and simple domain that covers both workspace and the obstacles
- Discard samples that lie in obstacles
- Link connectable samples with edges

Constituents

1. Generator of samples
2. Detector of obstacles
3. Local planner
4. Pseudo-metric $d(s_1, s_2) \in [0, \infty]$

$$\text{samples } s_1, s_2 \xrightarrow{\text{local planner}} \begin{cases} \text{either a way to move } s_1 \to s_2 \\ \text{or: failure} \end{cases}$$

Общая схема и компоненты

“First cover and then select” approach

- Build sampling of a large and simple domain that covers both workspace and the obstacles
- Discard samples that lie in obstacles
- Link connectable samples with edges

Constituents

1. Generator of samples
2. Detector of obstacles
3. Local planner
4. Pseudo-metric $d(s_1, s_2) \in [0, \infty]$

$$\text{samples } s_1, s_2 \xrightarrow{\text{local planner}} \begin{cases} \text{either a way to move } s_1 \to s_2 \\ \text{or: failure} \end{cases}$$

For a given sample s, invoke the local planner $LP(s, s_+)$ only for close enough destination samples s_+, i.e., such that $d(s, s_+) < \varepsilon$, where $\varepsilon > 0$ is a pre-specified constant

Общая схема и компоненты

“First cover and then select” approach

- Build sampling of a large and simple domain that covers both workspace and the obstacles
- Discard samples that lie in obstacles
- Link connectable samples with edges

Constituents

1. Generator of samples
2. Detector of obstacles
3. Local planner
4. Pseudo-metric $d(s_1, s_2) \in [0, \infty]$

samples $s_1, s_2 \xrightarrow{\text{local planner}} \begin{cases} \text{either a way to move } s_1 \to s_2 \\ \text{or: failure} \end{cases}$

For a given sample s, invoke the local planner $LP(s, s_+)$ only for close enough destination samples s_+, i.e., such that $d(s, s_+) < \varepsilon$, where $\varepsilon > 0$ is a pre-specified constant

Sampling set and sampling sequence

Sampling sequence $\Rightarrow$ sampling set
Algorithm of building a sampling sequence assumes an endless process

Dense sequence (in fact algorithm) $\overset{\text{def}}{\Leftrightarrow}$ this sequence is everywhere dense in the sampled set

Случайные плотные последовательности

Definition

A Borel probability measure $P(dx)$ on a metric space X is said to be complete if $P(U) > 0$ for any non-empty open subset $U \subset X$.

Случайные плотные последовательности

Definition

A Borel probability measure $P(dx)$ on a metric space X is said to be complete if $P(U) > 0$ for any non-empty open subset $U \subset X$.

Lemma

Suppose that a sequence $x_1, x_2, \ldots \in X$ is generated so that x_k is drawn independently of the previous samples $x_1, \ldots, x_{k-1}$ and in accordance with a complete probability distribution $P(dx)$. Then this sequence is dense with probability 1.

Случайные плотные последовательности

Definition

A Borel probability measure $P(dx)$ on a metric space X is said to be complete if $P(U) > 0$ for any non-empty open subset $U \subset X$.

Lemma

Suppose that a sequence $x_1, x_2, \ldots \in X$ is generated so that x_k is drawn independently of the previous samples $x_1, \ldots, x_{k-1}$ and in accordance with a complete probability distribution $P(dx)$. Then this sequence is dense with probability 1.

Proof: Let $x_* \in X$ and let U be an open vicinity of x_*. Then

$$P(U) > 0 \Rightarrow p := P(X \setminus U) = 1 - P(U) < 1$$

Случайные плотные последовательности

Definition

A Borel probability measure $P(dx)$ on a metric space X is said to be complete if $P(U) > 0$ for any non-empty open subset $U \subset X$.

Lemma

Suppose that a sequence $x_1, x_2, \ldots \in X$ is generated so that x_k is drawn independently of the previous samples $x_1, \ldots, x_{k-1}$ and in accordance with a complete probability distribution $P(dx)$. Then this sequence is dense with probability 1.

Proof: Let $x_* \in X$ and let U be an open vicinity of x_*. Then

$$P(U) > 0 \Rightarrow p := P(X \setminus U) = 1 - P(U) < 1$$

$$P[x_1 \notin U, \ldots, x_k \notin U] = \prod_{i=1}^{k} P[x_i \notin U] = p^k$$

Definition

A Borel probability measure $P(dx)$ on a metric space X is said to be complete if $P(U) > 0$ for any non-empty open subset $U \subset X$.

Lemma

Suppose that a sequence $x_1, x_2, \ldots \in X$ is generated so that x_k is drawn independently of the previous samples $x_1, \ldots, x_{k-1}$ and in accordance with a complete probability distribution $P(dx)$. Then this sequence is dense with probability 1.

Proof: Let $x_* \in X$ and let U be an open vicinity of x_*. Then

$$P(U) > 0 \Rightarrow p := P(X \setminus U) = 1 - P(U) < 1$$

$$P[x_1 \notin U, \ldots, x_k \notin U] = \prod_{i=1}^{k} P[x_i \notin U] = p^k$$

$$P[x_i \notin U \forall i \geq 1] \overset{\forall k}{\leq} P[x_1 \notin U, \ldots, x_k \notin U] = p^k \xrightarrow{k\to\infty} 0$$

Случайные плотные последовательности

Definition

A Borel probability measure $P(dx)$ on a metric space X is said to be complete if $P(U) > 0$ for any non-empty open subset $U \subset X$.

Lemma

Suppose that a sequence $x_1, x_2, \ldots \in X$ is generated so that x_k is drawn independently of the previous samples $x_1, \ldots, x_{k-1}$ and in accordance with a complete probability distribution $P(dx)$. Then this sequence is dense with probability 1.

Proof: Let $x_* \in X$ and let U be an open vicinity of x_*. Then

$$P(U) > 0 \Rightarrow p := P(X \setminus U) = 1 - P(U) < 1$$

$$P[x_1 \notin U, \ldots, x_k \notin U] = \prod_{i=1}^{k} P[x_i \notin U] = p^k$$

$$P[x_i \notin U \forall i \geq 1] \overset{\forall k}{\leq} P[x_1 \notin U, \ldots, x_k \notin U] = p^k \xrightarrow{k \to \infty} 0$$

Useful facts

- The Lebesgue measure is complete
- The Riemannian measure on a Riemannian manifold is complete
- Suppose that the probability measure $\mathcal{P}(dx)$ has a density $\rho(\cdot)$ with respect to a complete probability measure $P(dx)$ and $\rho(x) > 0$ for P-almost all $x \in X$. Then the measure $\mathcal{P}(dx)$ is complete.
- Suppose that for i=1,2, the probability measure $P_i(dx_i)$ is defined on the Borel σ-algebra of X_i and is complete. Then $P(dx_1) \otimes P_2(dx_2)$ is complete on the direct product $X_1 \times X_2$.

Случайные плотные последовательности

Definition

A Borel probability measure $P(dx)$ on a metric space X is said to be complete if $P(U) > 0$ for any non-empty open subset $U \subset X$.

Lemma

Suppose that a sequence $x_1, x_2, \ldots \in X$ is generated so that x_k is drawn independently of the previous samples $x_1, \ldots, x_{k-1}$ and in accordance with a complete probability distribution $P(dx)$. Then this sequence is dense with probability 1.

Proof: Let $x_* \in X$ and let U be an open vicinity of x_*. Then

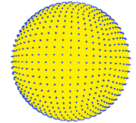
$$P(U) > 0 \Rightarrow p := P(X \setminus U) = 1 - P(U) < 1$$

$$P[x_1 \notin U, \ldots, x_k \notin U] = \prod_{i=1}^{k} P[x_i \notin U] = p^k$$

$$P[x_i \notin U \forall i \geq 1] \overset{\forall k}{\leq} P[x_1 \notin U, \ldots, x_k \notin U] = p^k \xrightarrow{k \to \infty} 0$$

Useful facts

- The Lebesgue measure is complete
- The Riemannian measure on a Riemannian manifold is complete
- Suppose that the probability measure $\mathcal{P}(dx)$ has a density $\rho(\cdot)$ with respect to a complete probability measure $P(dx)$ and $\rho(x) > 0$ for P-almost all $x \in X$. Then the measure $\mathcal{P}(dx)$ is complete.
- Suppose that for i=1,2, the probability measure $P_i(dx_i)$ is defined on the Borel σ-algebra of X_i and is complete. Then $P(dx_1) \otimes P_2(dx_2)$ is complete on the direct product $X_1 \times X_2$.



Случайные плотные последовательности

Definition

A Borel probability measure $P(dx)$ on a metric space X is said to be complete if $P(U) > 0$ for any non-empty open subset $U \subset X$.

Lemma

Suppose that a sequence $x_1, x_2, \ldots \in X$ is generated so that x_k is drawn independently of the previous samples $x_1, \ldots, x_{k-1}$ and in accordance with a complete probability distribution $P(dx)$. Then this sequence is dense with probability 1.

Proof: Let $x_* \in X$ and let U be an open vicinity of x_*. Then

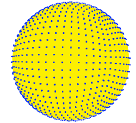
$$P(U) > 0 \Rightarrow p := P(X \setminus U) = 1 - P(U) < 1$$

$$P[x_1 \notin U, \ldots, x_k \notin U] = \prod_{i=1}^{k} P[x_i \notin U] = p^k$$

$$P[x_i \notin U \forall i \geq 1] \overset{\forall k}{\leq} P[x_1 \notin U, \ldots, x_k \notin U] = p^k \xrightarrow{k\to\infty} 0$$

Useful facts

- The Lebesgue measure is complete
- The Riemannian measure on a Riemannian manifold is complete
- Suppose that the probability measure $\mathcal{P}(dx)$ has a density $\rho(\cdot)$ with respect to a complete probability measure $P(dx)$ and $\rho(x) > 0$ for P-almost all $x \in X$. Then the measure $\mathcal{P}(dx)$ is complete.
- Suppose that for i=1,2, the probability measure $P_i(dx_i)$ is defined on the Borel σ-algebra of X_i and is complete. Then $P(dx_1) \otimes P_2(dx_2)$ is complete on the direct product $X_1 \times X_2$.



Pseudorandom number generators

- Linear congruential generator

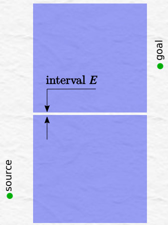
$x_k := m^{-1} X_k, X_{k+1} = (aX_k + c) \mod m$

- Lagged Fibonacci generator

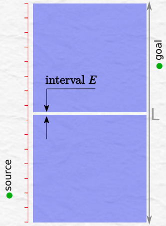
$$x_k := \begin{cases} x_{k-a} - x_{k-b} & \text{if } x_{k-a} \geq x_{k-b} \\ x_{k-a} - x_{k-b} + 1 & \text{otherwise} \end{cases}$$

Рандомность и плотность

Рандомность и плотность

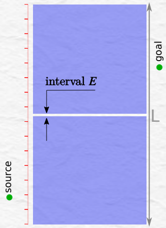


Рандомность и плотность



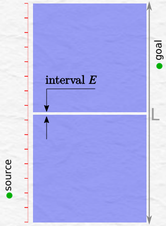
Рандомность и плотность

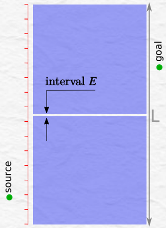
s – the number of the samples



Рандомность и плотность

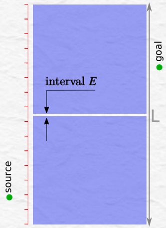
s – the number of the samples
feasible failure of even sampling $\frac{L}{s} > |E|$





s – the number of the samples
feasible failure of even sampling $\frac{L}{s} > |E|$
$x_1, \ldots, x_s$ random sampling of the interval of length L according to the uniform distribution

Рандомность и плотность

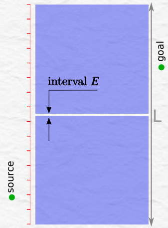


s – the number of the samples
feasible failure of even sampling $\frac{L}{s} > |E|$
$x_1, \ldots, x_s$ random sampling of the interval of length L according to the uniform distribution

$$\boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E] = \prod_{i=1}^{s} \boldsymbol{P}[x_i \notin E] = (1 - \boldsymbol{P}(E))^s$$

$$= \left(1 - \frac{|E|}{L}\right)^s \approx 1 - s\frac{|E|}{L}$$

Рандомность и плотность



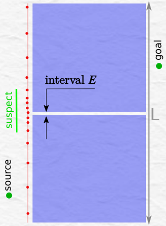
s – the number of the samples
feasible failure of even sampling $\frac{L}{s} > |E|$
$x_1, \ldots, x_s$ random sampling of the interval of length L according to the uniform distribution

$$\boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E] = \prod_{i=1}^{s} \boldsymbol{P}[x_i \notin E] = (1 - \boldsymbol{P}(E))^s$$

$$= \left(1 - \frac{|E|}{L}\right)^s \approx 1 - s\frac{|E|}{L}$$

$$\boldsymbol{P}[\exists i = 1, \ldots, s : x_i \in E] = 1 - \boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E]$$

$$= 1 - (1 - \boldsymbol{P}(E))^s \approx \frac{|E|}{L/s}$$



s – the number of the samples
feasible failure of even sampling $\frac{L}{s} > |E|$
$x_1, \ldots, x_s$ random sampling of the interval of length L according to the uniform distribution

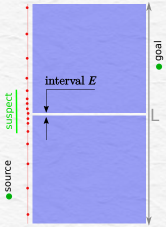
$$\boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E] = \prod_{i=1}^{s} \boldsymbol{P}[x_i \notin E] = (1 - \boldsymbol{P}(E))^s$$

$$= \left(1 - \frac{|E|}{L}\right)^s \approx 1 - s\frac{|E|}{L}$$

$$\boldsymbol{P}[\exists i = 1, \ldots, s : x_i \in E] = 1 - \boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E]$$

$$= 1 - (1 - \boldsymbol{P}(E))^s \approx \frac{|E|}{L/s}$$

Рандомность и плотность



s – the number of the samples

feasible failure of even sampling $\frac{L}{s} > |E|$

$x_1, \ldots, x_s$ random sampling of the interval of length L according to the $\boldsymbol{P}(dx)$

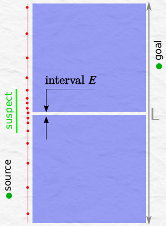
$$\boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E] = \prod_{i=1}^{s} \boldsymbol{P}[x_i \notin E] = (1 - \boldsymbol{P}(E))^s$$

$$= \left(1 - \frac{|E|}{L}\right)^s \approx 1 - s\frac{|E|}{L}$$

$$\boldsymbol{P}[\exists i = 1, \ldots, s : x_i \in E] = 1 - \boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E]$$

$$= 1 - (1 - \boldsymbol{P}(E))^s \approx \frac{|E|}{L/s}$$

Рандомность и плотность

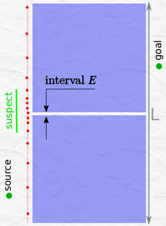


s – the number of the samples
feasible failure of even sampling $\frac{L}{s} > |E|$
$x_1, \ldots, x_s$ random sampling of the interval of length L according to the $\boldsymbol{P}(dx)$

$$\boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E] = \prod_{i=1}^{s} \boldsymbol{P}[x_i \notin E] = (1 - \boldsymbol{P}(E))^s$$

$$\boldsymbol{P}[\exists i = 1, \ldots, s : x_i \in E] = 1 - \boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E]$$
$$= 1 - (1 - \boldsymbol{P}(E))^s \to 1 \quad \text{as} \quad \boldsymbol{P}(E) \to 1$$

Рандомность и плотность



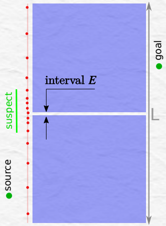
s – the number of the samples
feasible failure of even sampling $\frac{L}{s} > |E|$
$x_1, \dots, x_s$ random sampling of the interval of length L according to the $\boldsymbol{P}(dx)$

$$\boldsymbol{P}[x_1 \notin E, \dots, x_s \notin E] = \prod_{i=1}^{s} \boldsymbol{P}[x_i \notin E] = (1 - \boldsymbol{P}(E))^s$$

$$\boldsymbol{P}[\exists i = 1, \dots, s : x_i \in E] = 1 - \boldsymbol{P}[x_1 \notin E, \dots, x_s \notin E]$$
$$= 1 - (1 - \boldsymbol{P}(E))^s \to 1 \quad \text{as} \quad \boldsymbol{P}(E) \to 1$$

chi-squared test (χ^2): $X = (x_1, \dots, x_s) \sim \boldsymbol{P}(dx)$

- $X = A_1 \cup A_2 \cup \dots \cup A_k$ – partition of the sampled space



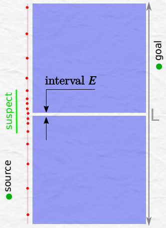
s – the number of the samples
feasible failure of even sampling $\frac{L}{s} > |E|$
$x_1, \dots, x_s$ random sampling of the interval of length L according to the $\boldsymbol{P}(dx)$

$$\boldsymbol{P}[x_1 \notin E, \dots, x_s \notin E] = \prod_{i=1}^{s} \boldsymbol{P}[x_i \notin E] = (1 - \boldsymbol{P}(E))^s$$

$$\boldsymbol{P}[\exists i = 1, \dots, s : x_i \in E] = 1 - \boldsymbol{P}[x_1 \notin E, \dots, x_s \notin E]$$
$$= 1 - (1 - \boldsymbol{P}(E))^s \to 1 \quad \text{as} \quad \boldsymbol{P}(E) \to 1$$

chi-squared test (χ^2): $X = (x_1, \dots, x_s) \sim \boldsymbol{P}(dx)$

- $X = A_1 \cup A_2 \cup \dots \cup A_k$ – partition of the sampled space
- $E_j := s\boldsymbol{P}(A_j)$ – theoretically expected number of samples in the set A_j



s – the number of the samples
feasible failure of even sampling $\frac{L}{s} > |E|$
$x_1, \ldots, x_s$ random sampling of the interval of length L according to the $\boldsymbol{P}(dx)$

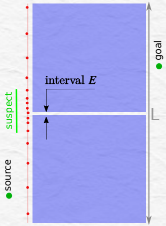
$$\boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E] = \prod_{i=1}^{s} \boldsymbol{P}[x_i \notin E] = (1 - \boldsymbol{P}(E))^s$$

$$\boldsymbol{P}[\exists i = 1, \ldots, s : x_i \in E] = 1 - \boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E]$$
$$= 1 - (1 - \boldsymbol{P}(E))^s \to 1 \quad \text{as} \quad \boldsymbol{P}(E) \to 1$$

chi-squared test (χ^2): $X = (x_1, \ldots, x_s) \sim \boldsymbol{P}(dx)$

- $X = A_1 \cup A_2 \cup \ldots \cup A_k$ – partition of the sampled space
- $E_j := s\boldsymbol{P}(A_j)$ – theoretically expected number of samples in the set A_j
- $N_j := |\{i : x_i \in A_j\}|$ actually observed number of samples

Рандомность и плотность



s – the number of the samples
feasible failure of even sampling $\frac{L}{s} > |E|$
$x_1, \dots, x_s$ random sampling of the interval of length L according to the $\boldsymbol{P}(dx)$

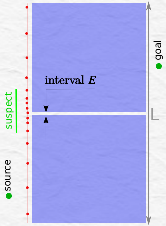
$$\boldsymbol{P}[x_1 \notin E, \dots, x_s \notin E] = \prod_{i=1}^{s} \boldsymbol{P}[x_i \notin E] = (1 - \boldsymbol{P}(E))^s$$

$$\boldsymbol{P}[\exists i = 1, \dots, s : x_i \in E] = 1 - \boldsymbol{P}[x_1 \notin E, \dots, x_s \notin E]$$
$$= 1 - (1 - \boldsymbol{P}(E))^s \to 1 \quad \text{as} \quad \boldsymbol{P}(E) \to 1$$

chi-squared test (χ^2): $X = (x_1, \dots, x_s) \sim \boldsymbol{P}(dx)$

- $X = A_1 \cup A_2 \cup \dots \cup A_k$ – partition of the sampled space
- $E_j := s\boldsymbol{P}(A_j)$ – theoretically expected number of samples in the set A_j
- $N_j := |\{i : x_i \in A_j\}|$ actually observed number of samples
- $\chi^2(X) := \sum_{j=1}^{k} \frac{(N_j - E_j)^2}{E_j}$

Рандомность и плотность



s – the number of the samples
feasible failure of even sampling $\frac{L}{s} > |E|$
$x_1, \ldots, x_s$ random sampling of the interval of length L according to the $\boldsymbol{P}(dx)$

$$\boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E] = \prod_{i=1}^{s} \boldsymbol{P}[x_i \notin E] = (1 - \boldsymbol{P}(E))^s$$

$$\boldsymbol{P}[\exists i = 1, \ldots, s : x_i \in E] = 1 - \boldsymbol{P}[x_1 \notin E, \ldots, x_s \notin E]$$
$$= 1 - (1 - \boldsymbol{P}(E))^s \to 1 \quad \text{as} \quad \boldsymbol{P}(E) \to 1$$

chi-squared test (χ^2): $X = (x_1, \ldots, x_s) \sim \boldsymbol{P}(dx)$

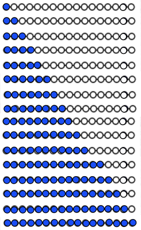
- $X = A_1 \cup A_2 \cup \ldots \cup A_k$ – partition of the sampled space
- $E_j := s\boldsymbol{P}(A_j)$ – theoretically expected number of samples in the set A_j
- $N_j := |\{i : x_i \in A_j\}|$ actually observed number of samples
- $\chi^2(X) := \sum_{j=1}^{k} \frac{(N_j - E_j)^2}{E_j}$

Test

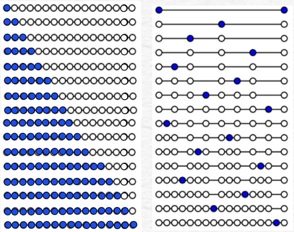
$$\chi_1^2 \le \chi^2(X) \le \chi_2^2$$

Последовательность ван дер Корпута

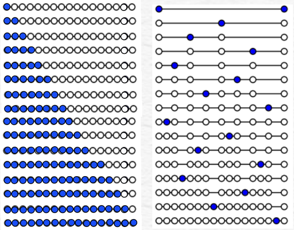
Последовательность ван дер Корпута



Последовательность ван дер Корпута

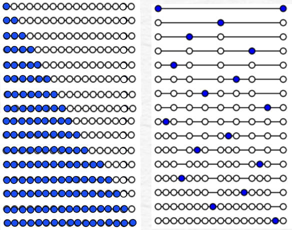


Последовательность ван дер Корпута



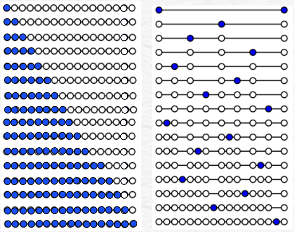
Van der Corput sequence

Последовательность ван дер Корпута



Van der Corput sequence

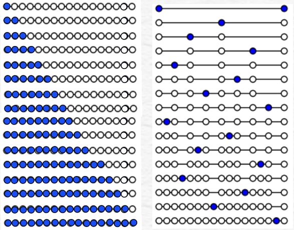
- b – integer, the base of a numeral system



Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$

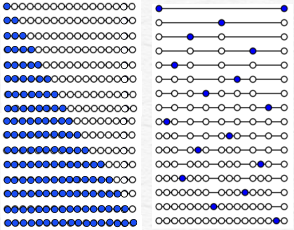
Последовательность ван дер Корпута



Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0, 1]$ - van der Corput sequence

Последовательность ван дер Корпута

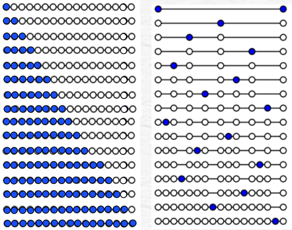


Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0, 1]$ - van der Corput sequence

Discrepancy with the probability distribution with respect to a class of sets

Последовательность ван дер Корпута



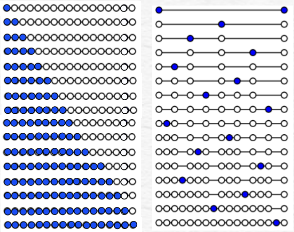
Van der Corput sequence

- b – integer, the base of a numeral system
- index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
- $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0, 1]$ - van der Corput sequence

Discrepancy with the probability distribution with respect to a class of sets

- $\boldsymbol{P}(dx)$ – Borel probability distribution on a metric space

Последовательность ван дер Корпута

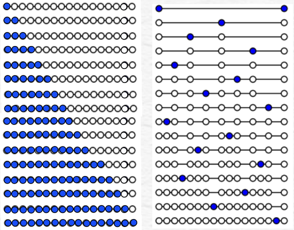


Van der Corput sequence

- b – integer, the base of a numeral system
- index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
- $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0, 1]$ - van der Corput sequence

Discrepancy with the probability distribution with respect to a class of sets

- $\boldsymbol{P}(dx)$ – Borel probability distribution on a metric space
- $\mathcal{R}$ – a collection of sets, each being Borel



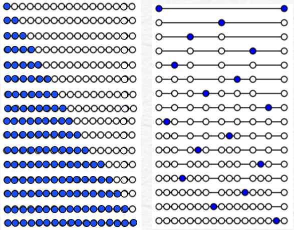
Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0,1]$ - van der Corput sequence

Discrepancy with the probability distribution with respect to a class of sets

- $\boldsymbol{P}(dx)$ – Borel probability distribution on a metric space
  - $\mathcal{R}$ – a collection of sets, each being Borel
  - $X = (x_1, \ldots, x_s)$ – a sequence of space points

Последовательность ван дер Корпута



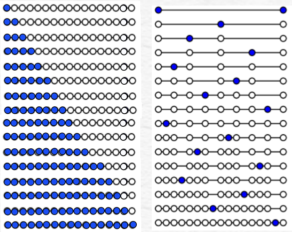
Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0, 1]$ - van der Corput sequence

Discrepancy with the probability distribution with respect to a class of sets

- $P(dx)$ – Borel probability distribution on a metric space
  - $\mathcal{R}$ – a collection of sets, each being Borel
  - $X = (x_1, \ldots, x_s)$ – a sequence of space points
  - $N_X(A) := |\{i : x_i \in A\}|$ – the number of samples in A

Последовательность ван дер Корпута



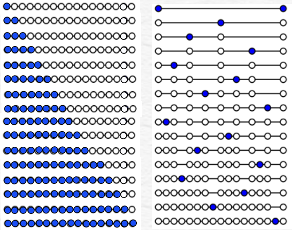
Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0,1]$ - van der Corput sequence

Discrepancy with the probability distribution with respect to a class of sets

- $\boldsymbol{P}(dx)$ – Borel probability distribution on a metric space
  - $\mathfrak{R}$ – a collection of sets, each being Borel
  - $X = (x_1, \ldots, x_s)$ – a sequence of space points
  - $N_X(A) := |\{i : x_i \in A\}|$ – the number of samples in A
  - $$D_{\mathfrak{R}}\left[X \sim \boldsymbol{P}(dx)\right] := \sup_{A \in \mathfrak{R}} \left| \frac{N_X(A)}{s} - \boldsymbol{P}(A) \right|$$

Последовательность ван дер Корпута



Van der Corput sequence

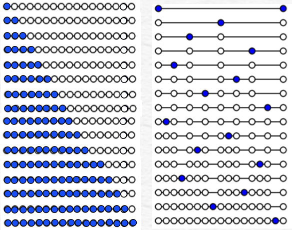
- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0, 1]$ - van der Corput sequence

Discrepancy with the probability distribution with respect to a class of sets

- $\boldsymbol{P}(dx)$ – Borel probability distribution on a metric space
  - $\mathfrak{R}$ – a collection of sets, each being Borel
  - $X = (x_1, \ldots, x_s)$ – a sequence of space points
  - $N_X(A) := |\{i : x_i \in A\}|$ – the number of samples in A
  - $$D_{\mathfrak{R}}\left[X \sim \boldsymbol{P}(dx)\right] := \sup_{A \in \mathfrak{R}} \left| \frac{N_X(A)}{s} - \boldsymbol{P}(A) \right|$$

Popular choices of the class of sets

Последовательность ван дер Корпута



Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0, 1]$ - van der Corput sequence

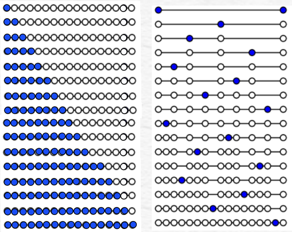
Discrepancy with the probability distribution with respect to a class of sets

- $\boldsymbol{P}(dx)$ – Borel probability distribution on a metric space
  - $\mathcal{R}$ – a collection of sets, each being Borel
  - $X = (x_1, \ldots, x_s)$ – a sequence of space points
  - $N_X(A) := |\{i : x_i \in A\}|$ – the number of samples in A
  - $$D_{\mathcal{R}}[X \sim \boldsymbol{P}(dx)] := \sup_{A \in \mathcal{R}} \left| \frac{N_X(A)}{s} - \boldsymbol{P}(A) \right|$$

Popular choices of the class of sets

- on the real line: $\mathcal{R}$ consists of all intervals

Последовательность ван дер Корпута



Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0, 1]$ - van der Corput sequence

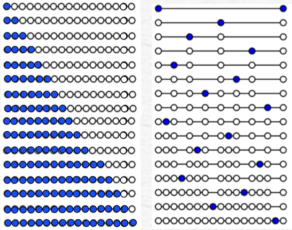
Discrepancy with the probability distribution with respect to a class of sets

- $\boldsymbol{P}(dx)$ – Borel probability distribution on a metric space
  - $\mathcal{R}$ – a collection of sets, each being Borel
  - $X = (x_1, \ldots, x_s)$ – a sequence of space points
  - $N_X(A) := |\{i : x_i \in A\}|$ – the number of samples in A
  - $$D_{\mathcal{R}}\left[X \sim \boldsymbol{P}(dx)\right] := \sup_{A \in \mathcal{R}} \left| \frac{N_X(A)}{s} - \boldsymbol{P}(A) \right|$$

Popular choices of the class of sets

- on the real line: $\mathcal{R}$ consists of all intervals
- on $[a, b] \subset \mathbb{R}$: the set $\mathcal{R} = \{[a, x] : x \in [a, b]\} \sim D^*$

Последовательность ван дер Корпута



Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0, 1]$ - van der Corput sequence

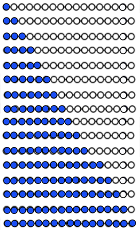
Discrepancy with the probability distribution with respect to a class of sets

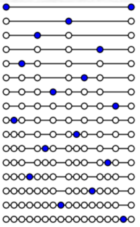
- $\boldsymbol{P}(dx)$ – Borel probability distribution on a metric space
  - $\mathcal{R}$ – a collection of sets, each being Borel
  - $X = (x_1, \ldots, x_s)$ – a sequence of space points
  - $N_X(A) := |\{i : x_i \in A\}|$ – the number of samples in A
  - $$D_{\mathcal{R}}\left[X \sim \boldsymbol{P}(dx)\right] := \sup_{A \in \mathcal{R}} \left| \frac{N_X(A)}{s} - \boldsymbol{P}(A) \right|$$

Popular choices of the class of sets

- on the real line: $\mathcal{R}$ consists of all intervals
- on $[a, b] \subset \mathbb{R}$: the set $\mathcal{R} = \{[a, x] : x \in [a, b]\} \sim D^*$
- in $\mathbb{R}^n$: $\mathcal{R} \leftrightarrow$ all hyper-parallelepipeds $\prod_{i=1}^n [a_i, b_i]$

Последовательность ван дер Корпута





Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0,1]$ - van der Corput sequence

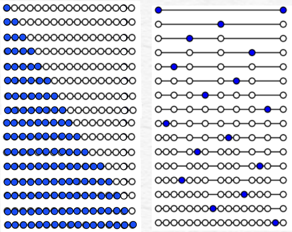
Discrepancy with the probability distribution with respect to a class of sets

- $\boldsymbol{P}(dx)$ – Borel probability distribution on a metric space
  - $\mathcal{R}$ – a collection of sets, each being Borel
  - $X = (x_1, \ldots, x_s)$ – a sequence of space points
  - $N_X(A) := |\{i : x_i \in A\}|$ – the number of samples in A
  - $$D_{\mathcal{R}}\left[X \sim \boldsymbol{P}(dx)\right] := \sup_{A \in \mathcal{R}} \left| \frac{N_X(A)}{s} - \boldsymbol{P}(A) \right|$$

Popular choices of the class of sets

- on the real line: $\mathcal{R}$ consists of all intervals
- on $[a,b] \subset \mathbb{R}$: the set $\mathcal{R} = \{[a,x] : x \in [a,b]\} \sim D^*$
- in $\mathbb{R}^n$: $\mathcal{R} \leftrightarrow$ all hyper-parallelepipeds $\prod_{i=1}^n [a_i, b_i]$
- in a hyper-parallelepiped $\prod_{i=1}^n [a_i, b_i]$: the set $\mathcal{R} = \{\prod_{i=1}^n [a_i, x_i] : x_i \in [a_i, b_i]\} \sim D^*$

Последовательность ван дер Корпута



Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0,1]$ - van der Corput sequence

Discrepancy with the probability distribution with respect to a class of sets

- $\boldsymbol{P}(dx)$ – Borel probability distribution on a metric space
  - $\mathcal{R}$ – a collection of sets, each being Borel
  - $X = (x_1, \dots, x_s)$ – a sequence of space points
  - $N_X(A) := |\{i : x_i \in A\}|$ – the number of samples in A
  - $$D_{\mathcal{R}}\left[X \sim \boldsymbol{P}(dx)\right] := \sup_{A \in \mathcal{R}} \left| \frac{N_X(A)}{s} - \boldsymbol{P}(A) \right|$$

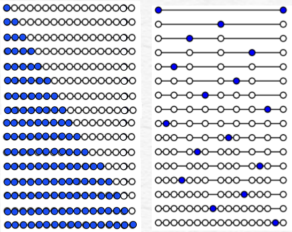
Popular choices of the class of sets

- on the real line: $\mathcal{R}$ consists of all intervals
- on $[a,b] \subset \mathbb{R}$: the set $\mathcal{R} = \{[a,x] : x \in [a,b]\} \sim D^*$
- in $\mathbb{R}^n$: $\mathcal{R} \leftrightarrow$ all hyper-parallelepipeds $\prod_{i=1}^n [a_i, b_i]$
- in a hyper-parallelepiped $\prod_{i=1}^n [a_i, b_i]$: the set $\mathcal{R} = \{\prod_{i=1}^n [a_i, x_i] : x_i \in [a_i, b_i]\} \sim D^*$

Useful facts

- in $\mathbb{R}^d$: we have $D^*_{\mathcal{R}} \leq D_{\mathcal{R}} \leq 2^d D^*_{\mathcal{R}}$

Последовательность ван дер Корпута



Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0,1]$ - van der Corput sequence

Discrepancy with the probability distribution with respect to a class of sets

- $\boldsymbol{P}(dx)$ – Borel probability distribution on a metric space
  - $\mathcal{R}$ – a collection of sets, each being Borel
  - $X = (x_1, \ldots, x_s)$ – a sequence of space points
  - $N_X(A) := |\{i : x_i \in A\}|$ – the number of samples in A
  - $$D_{\mathcal{R}}\left[X \sim \boldsymbol{P}(dx)\right] := \sup_{A \in \mathcal{R}} \left| \frac{N_X(A)}{s} - \boldsymbol{P}(A) \right|$$

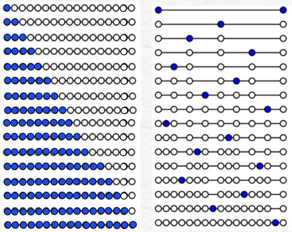
Popular choices of the class of sets

- on the real line: $\mathcal{R}$ consists of all intervals
- on $[a,b] \subset \mathbb{R}$: the set $\mathcal{R} = \{[a,x] : x \in [a,b]\} \sim D^*$
- in $\mathbb{R}^n$: $\mathcal{R} \leftrightarrow$ all hyper-parallelepipeds $\prod_{i=1}^n [a_i, b_i]$
- in a hyper-parallelepiped $\prod_{i=1}^n [a_i, b_i]$: the set $\mathcal{R} = \{\prod_{i=1}^n [a_i, x_i] : x_i \in [a_i, b_i]\} \sim D^*$

Useful facts

- in $\mathbb{R}^d$: we have $D^*_{\mathcal{R}} \le D_{\mathcal{R}} \le 2^d D^*_{\mathcal{R}}$
- the uniform distribution: $D^*_{\mathcal{R}} \ge \frac{1}{n}$ for $d = 1$ and $D^*_{\mathcal{R}} \ge C \frac{\log n}{n}$ for $d = 2$

Последовательность ван дер Корпута



Van der Corput sequence

- b – integer, the base of a numeral system
  - index $k = \sum_{i=0}^{f-1} \varsigma_i(k) b^i$, where $\varsigma_i(k) \in [0 : b-1]$
  - $x_k = \sum_{i=0}^{f-1} \varsigma_i(k) b^{-i-1} \in [0,1]$ - van der Corput sequence

Discrepancy with the probability distribution with respect to a class of sets

- $\boldsymbol{P}(dx)$ – Borel probability distribution on a metric space
  - $\mathcal{R}$ – a collection of sets, each being Borel
  - $X = (x_1, \ldots, x_s)$ – a sequence of space points
  - $N_X(A) := |\{i : x_i \in A\}|$ – the number of samples in A
  - $$D_{\mathcal{R}}\left[X \sim \boldsymbol{P}(dx)\right] := \sup_{A \in \mathcal{R}} \left| \frac{N_X(A)}{s} - \boldsymbol{P}(A) \right|$$

Popular choices of the class of sets

- on the real line: $\mathcal{R}$ consists of all intervals
- on $[a,b] \subset \mathbb{R}$: the set $\mathcal{R} = \{[a,x] : x \in [a,b]\} \sim D^*$
- in $\mathbb{R}^n$: $\mathcal{R} \leftrightarrow$ all hyper-parallelepipeds $\prod_{i=1}^n [a_i, b_i]$
- in a hyper-parallelepiped $\prod_{i=1}^n [a_i, b_i]$: the set $\mathcal{R} = \{\prod_{i=1}^n [a_i, x_i] : x_i \in [a_i, b_i]\} \sim D^*$

Useful facts

- in $\mathbb{R}^d$: we have $D^*_{\mathcal{R}} \leq D_{\mathcal{R}} \leq 2^d D^*_{\mathcal{R}}$
- the uniform distribution: $D^*_{\mathcal{R}} \geq \frac{1}{n}$ for $d = 1$ and $D^*_{\mathcal{R}} \geq C \frac{\log n}{n}$ for $d = 2$
- van der Corput: $D^*_{\mathcal{R}} \leq K \frac{\log n}{n}$

Многомерный случай: последовательность Холтона и множество Хаммерсли

Многомерный случай: последовательность Холтона и множество Хаммерсли

Halton sequence in $\mathbb{R}^n$

- $b_1, \ldots, b_n$ – relatively prime integers (usually the first n primes)

Многомерный случай: последовательность Холтона и множество Хаммерсли

Halton sequence in $\mathbb{R}^n$

- $b_1, \ldots, b_n$ – relatively prime integers (usually the first n primes)
- index $k = \varsigma_0(k|i)b_i^0 + \varsigma_1(k|i)b_i^1 + \varsigma_2(k|i)b_i^2 + \cdots$ – representation of k in the b_i-based numeral system

Многомерный случай: последовательность Холтона и множество Хаммерсли

Halton sequence in $\mathbb{R}^n$

- $b_1, \dots, b_n$ – relatively prime integers (usually the first n primes)
- index $k = \varsigma_0(k|i)b_i^0 + \varsigma_1(k|i)b_i^1 + \varsigma_2(k|i)b_i^2 + \cdots$ – representation of k in the b_i-based numeral system
- $x(k|i) = \varsigma_0(k|i)b_i^{-1} + \varsigma_1(k|i)b_i^{-2} + \varsigma_2(k|i)b_i^{-3} + \cdots \in [0, 1]$

Многомерный случай: последовательность Холтона и множество Хаммерсли

Halton sequence in $\mathbb{R}^n$

- $b_1, \dots, b_n$ – relatively prime integers (usually the first n primes)
- index $k = \varsigma_0(k|i)b_i^0 + \varsigma_1(k|i)b_i^1 + \varsigma_2(k|i)b_i^2 + \cdots$ – representation of k in the b_i-based numeral system
- $x(k|i) = \varsigma_0(k|i)b_i^{-1} + \varsigma_1(k|i)b_i^{-2} + \varsigma_2(k|i)b_i^{-3} + \cdots \in [0, 1]$
- $x(k) = \left[x(k|1), x(k|2), \dots, x(k|n)\right] \in [0, 1]^n, k = 1, 2, \dots$ – Halton sequence

Многомерный случай: последовательность Холтона и множество Хаммерсли

Halton sequence in $\mathbb{R}^n$

- $b_1, \ldots, b_n$ – relatively prime integers (usually the first n primes)
- index $k = \varsigma_0(k|i)b_i^0 + \varsigma_1(k|i)b_i^1 + \varsigma_2(k|i)b_i^2 + \cdots$ – representation of k in the b_i-based numeral system
- $x(k|i) = \varsigma_0(k|i)b_i^{-1} + \varsigma_1(k|i)b_i^{-2} + \varsigma_2(k|i)b_i^{-3} + \cdots \in [0, 1]$
- $x(k) = \left[x(k|1), x(k|2), \ldots, x(k|n)\right] \in [0, 1]^n, k = 1, 2, \ldots$ – Halton sequence

Hammersley point set (the number k of samples is given)

$$x(j) = \left[\frac{j}{k}, x(k|1), x(k|2), \ldots, x(k|n-1)\right] \in [0, 1]^n, j = 1, 2, \ldots, k$$

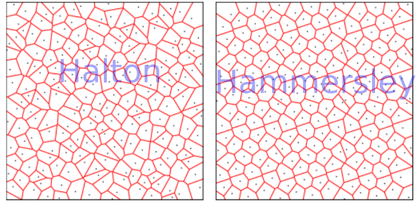
Многомерный случай: последовательность Холтона и множество Хаммерсли

Halton sequence in $\mathbb{R}^n$

- $b_1, \ldots, b_n$ – relatively prime integers (usually the first n primes)
- index $k = \varsigma_0(k|i)b_i^0 + \varsigma_1(k|i)b_i^1 + \varsigma_2(k|i)b_i^2 + \cdots$ – representation of k in the b_i-based numeral system
- $x(k|i) = \varsigma_0(k|i)b_i^{-1} + \varsigma_1(k|i)b_i^{-2} + \varsigma_2(k|i)b_i^{-3} + \cdots \in [0, 1]$
- $x(k) = [x(k|1), x(k|2), \ldots, x(k|n)] \in [0, 1]^n, k = 1, 2, \ldots$ – Halton sequence

Hammersley point set (the number k of samples is given)

$$x(j) = \left[\frac{j}{k}, x(k|1), x(k|2), \ldots, x(k|n-1)\right] \in [0, 1]^n, j = 1, 2, \ldots, k$$



Сэмплирование с низкой дисперсией; сетки, решетки.

Сэмплирование с низкой дисперсией; сетки, решетки.

Dispersion

Сэмплирование с низкой дисперсией; сетки, решетки.

Dispersion

- X is a metric space with the distance function $d(\cdot, \cdot)$

Сэмплирование с низкой дисперсией; сетки, решетки.

Dispersion

- X is a metric space with the distance function $d(\cdot,\cdot)$
- $S \subset X$ is a subset (typically finite)

Dispersion

- X is a metric space with the distance function $d(\cdot, \cdot)$
- $S \subset X$ is a subset (typically finite)
- Dispersion of the set S is defined to be $\delta(S) := \sup_{x \in X} \inf_{s \in S} d(x, s)$

Сэмплирование с низкой дисперсией; сетки, решетки.

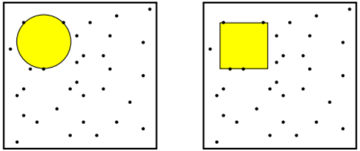
Dispersion

- X is a metric space with the distance function $d(\cdot,\cdot)$
- $S \subset X$ is a subset (typically finite)
- Dispersion of the set S is defined to be $\delta(S) := \sup_{x\in X} \inf_{s\in S} d(x,s)$
- This is the maximal radius of an open ball that has no points in common with S

Сэмплирование с низкой дисперсией; сетки, решетки.

Dispersion

- X is a metric space with the distance function $d(\cdot,\cdot)$
- $S \subset X$ is a subset (typically finite)
- Dispersion of the set S is defined to be $\delta(S) := \sup_{x\in X} \inf_{s\in S} d(x,s)$
- This is the maximal radius of an open ball that has no points in common with S

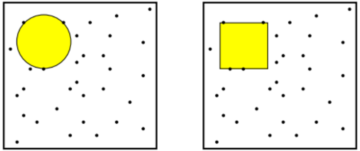


(a) L_2 dispersion (b) L_∞ dispersion

Сэмплирование с низкой дисперсией; сетки, решетки.

Dispersion

- X is a metric space with the distance function $d(\cdot,\cdot)$
- $S \subset X$ is a subset (typically finite)
- Dispersion of the set S is defined to be $\delta(S) := \sup_{x\in X} \inf_{s\in S} d(x,s)$
- This is the maximal radius of an open ball that has no points in common with S



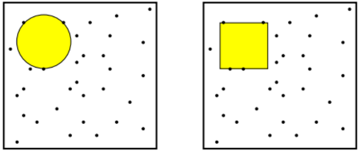
(a) L_2 dispersion (b) L_∞ dispersion

$X = [0,1]^n$, L_∞-metric

Сэмплирование с низкой дисперсией; сетки, решетки.

Dispersion

- X is a metric space with the distance function $d(\cdot,\cdot)$
- $S \subset X$ is a subset (typically finite)
- Dispersion of the set S is defined to be $\delta(S) := \sup_{x\in X} \inf_{s\in S} d(x,s)$
- This is the maximal radius of an open ball that has no points in common with S



(a) L_2 dispersion (b) L_∞ dispersion

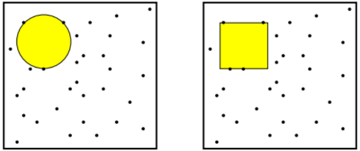
$X = [0,1]^n$, L_∞-metric

- The number k of samples is given

Сэмплирование с низкой дисперсией; сетки, решетки.

Dispersion

- X is a metric space with the distance function $d(\cdot,\cdot)$
- $S \subset X$ is a subset (typically finite)
- Dispersion of the set S is defined to be $\delta(S) := \sup_{x \in X} \inf_{s \in S} d(x,s)$
- This is the maximal radius of an open ball that has no points in common with S



(a) L_2 dispersion (b) L_∞ dispersion

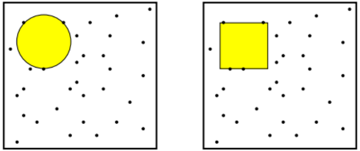
$X = [0,1]^n$, L_∞-metric

- The number k of samples is given
- The best solution (Sukharev's grid) - partition of the sampled cube $[0,1]^n$ into smaller cubes with a common side-length l, whose centers are samples

Сэмплирование с низкой дисперсией; сетки, решетки.

Dispersion

- X is a metric space with the distance function $d(\cdot,\cdot)$
- $S \subset X$ is a subset (typically finite)
- Dispersion of the set S is defined to be $\delta(S) := \sup_{x\in X} \inf_{s\in S} d(x,s)$
- This is the maximal radius of an open ball that has no points in common with S



(a) L_2 dispersion (b) L_∞ dispersion

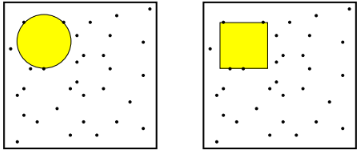
$X = [0,1]^n$, L_∞-metric

- The number k of samples is given
- The best solution (Sukharev's grid) - partition of the sampled cube $[0,1]^n$ into smaller cubes with a common side-length l, whose centers are samples
- Specifically, $l = \left\lfloor k^{\frac{1}{n}} \right\rfloor^{-1}$

Сэмплирование с низкой дисперсией; сетки, решетки.

Dispersion

- X is a metric space with the distance function $d(\cdot,\cdot)$
- $S \subset X$ is a subset (typically finite)
- Dispersion of the set S is defined to be $\delta(S) := \sup_{x\in X}\inf_{s\in S} d(x,s)$
- This is the maximal radius of an open ball that has no points in common with S



(a) L_2 dispersion (b) L_∞ dispersion

$X=[0,1]^n$, L_∞-metric

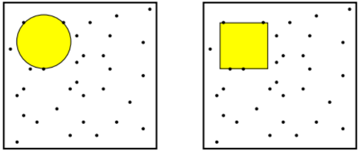
- The number k of samples is given
- The best solution (Sukharev's grid) - partition of the sampled cube $[0,1]^n$ into smaller cubes with a common side-length l, whose centers are samples
- Specifically, $l = \left\lfloor k^{\frac{1}{n}} \right\rfloor^{-1}$

$X=[0,1]^2$, L_2-metric

Сэмплирование с низкой дисперсией; сетки, решетки.

Dispersion

- X is a metric space with the distance function $d(\cdot,\cdot)$
- $S \subset X$ is a subset (typically finite)
- Dispersion of the set S is defined to be $\delta(S) := \sup_{x\in X} \inf_{s\in S} d(x,s)$
- This is the maximal radius of an open ball that has no points in common with S



(a) L_2 dispersion (b) L_∞ dispersion

$X = [0,1]^n$, L_∞-metric

- The number k of samples is given
- The best solution (Sukharev's grid) - partition of the sampled cube $[0,1]^n$ into smaller cubes with a common side-length l, whose centers are samples
- Specifically, $l = \left\lfloor k^{\frac{1}{n}} \right\rfloor^{-1}$

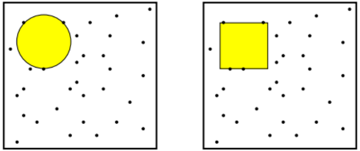
$X = [0,1]^2$, L_2-metric

- The number k of samples is given

Сэмплирование с низкой дисперсией; сетки, решетки.

Dispersion

- X is a metric space with the distance function $d(\cdot,\cdot)$
- $S \subset X$ is a subset (typically finite)
- Dispersion of the set S is defined to be $\delta(S) := \sup_{x\in X} \inf_{s\in S} d(x,s)$
- This is the maximal radius of an open ball that has no points in common with S



(a) L_2 dispersion (b) L_∞ dispersion

$X = [0,1]^n$, L_∞-metric

- The number k of samples is given
- The best solution (Sukharev's grid) - partition of the sampled cube $[0,1]^n$ into smaller cubes with a common side-length l, whose centers are samples
- Specifically, $l = \left\lfloor k^{\frac{1}{n}} \right\rfloor^{-1}$

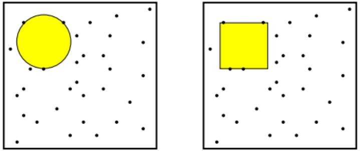
$X = [0,1]^2$, L_2-metric

- The number k of samples is given
- The best solution is tiling with equilateral triangles

Сэмплирование с низкой дисперсией; сетки, решетки.

Dispersion

- X is a metric space with the distance function $d(\cdot,\cdot)$
- $S \subset X$ is a subset (typically finite)
- Dispersion of the set S is defined to be $\delta(S) := \sup_{x\in X} \inf_{s\in S} d(x,s)$
- This is the maximal radius of an open ball that has no points in common with S



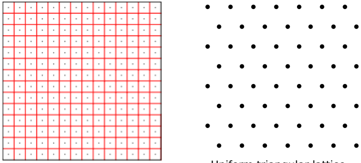
(a) L_2 dispersion (b) L_∞ dispersion

$X = [0,1]^n$, L_∞-metric

- The number k of samples is given
- The best solution (Sukharev's grid) - partition of the sampled cube $[0,1]^n$ into smaller cubes with a common side-length l, whose centers are samples
- Specifically, $l = \left\lfloor k^{\frac{1}{n}} \right\rfloor^{-1}$

$X = [0,1]^2$, L_2-metric

- The number k of samples is given
- The best solution is tiling with equilateral triangles



(a) 196-point Sukharev grid

Uniform triangular lattice

Политика по отношению точкам из запретных областей

- Ignoring (neglecting, skipping, missing)
- Using to build a sample in the free space

Политика по отношению точкам из запретных областей

- Ignoring (neglecting, skipping, missing)
- Using to build a sample in the free space

Local planner for a given free space

A device (algorithm) that is fed by a pair of samples (s_0, s_+) and outputs the following:

Политика по отношению точкам из запретных областей

- Ignoring (neglecting, skipping, missing)
- Using to build a sample in the free space

Local planner for a given free space

A device (algorithm) that is fed by a pair of samples (s_0, s_+) and outputs the following:

- Either a path $\mathfrak{P}$ from s_0 to s_+ that goes through the given free space and can be traced by the robot (robots)

Политика по отношению точкам из запретных областей

- Ignoring (neglecting, skipping, missing)
- Using to build a sample in the free space

Local planner for a given free space

A device (algorithm) that is fed by a pair of samples (s_0, s_+) and outputs the following:

- Either a path $\mathfrak{P}$ from s_0 to s_+ that goes through the given free space and can be traced by the robot (robots)
- Or a special symbol ✠ that signal about a failure to build $\mathfrak{P}$

Политика по отношению точкам из запретных областей

- Ignoring (neglecting, skipping, missing)
- Using to build a sample in the free space

Local planner for a given free space

A device (algorithm) that is fed by a pair of samples (s_0, s_+) and outputs the following:

- Either a path $\mathfrak{P}$ from s_0 to s_+ that goes through the given free space and can be traced by the robot (robots)
- Or a special symbol ✠ that signal about a failure to build $\mathfrak{P}$

The local planner is said to be locally complete if there exists $\varepsilon > 0$ such that the output $\neq$ ✠ whenever the ball with a radius of ε centered at s_0 fully lies in the free space and contains s_+.

Политика по отношению точкам из запретных областей

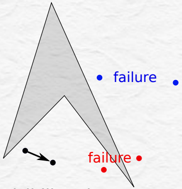
- Ignoring (neglecting, skipping, missing)
- Using to build a sample in the free space

Local planner for a given free space

A device (algorithm) that is fed by a pair of samples (s_0, s_+) and outputs the following:

- Either a path $\mathfrak{P}$ from s_0 to s_+ that goes through the given free space and can be traced by the robot (robots)
- Or a special symbol ✠ that signal about a failure to build $\mathfrak{P}$

The local planner is said to be locally complete if there exists $\varepsilon > 0$ such that the output $\neq$ ✠ whenever the ball with a radius of ε centered at s_0 fully lies in the free space and contains s_+.



visibility planner

Политика по отношению точкам из запретных областей

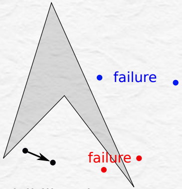
- Ignoring (neglecting, skipping, missing)
- Using to build a sample in the free space

Local planner for a given free space

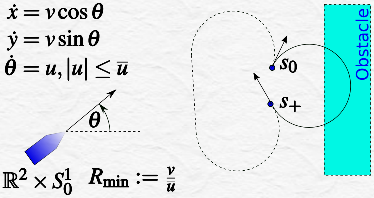
A device (algorithm) that is fed by a pair of samples (s_0, s_+) and outputs the following:

- Either a path $\mathfrak{P}$ from s_0 to s_+ that goes through the given free space and can be traced by the robot (robots)
- Or a special symbol $\maltese$ that signal about a failure to build $\mathfrak{P}$

The local planner is said to be locally complete if there exists $\varepsilon > 0$ such that the output $\neq \maltese$ whenever the ball with a radius of ε centered at s_0 fully lies in the free space and contains s_+.



visibility planner



non-holonomic local planner

Политика по отношению точкам из запретных областей

- Ignoring (neglecting, skipping, missing)
- Using to build a sample in the free space

Using local planner $\mathfrak{LP}$ for the ideal obstacle-free space

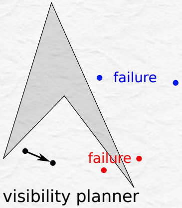
- Form the set S_{true} of "true" samples step-by-step
- If a new sample s_+ lies in the free space, add this sample to S_{true}
- Otherwise
  - Find a sample $s_0 \in S_{\mathrm{true}}$ most beneficial for $\mathfrak{LP}$ to build a path $\mathcal{P}$ from s_0 to s_+ in the ideal, obstacle-free space (e.g., the point s_0 nearest to s_+)
  - Run $\mathfrak{LP}$ to build $\mathcal{P}$
  - Truncate $\mathcal{P}$ by leaving some its initial portion $\mathcal{P}_{\mathrm{in}}$ that does not collide with the obstacles
  - Enrich S_{true} with the end of $\mathcal{P}_{\mathrm{in}}$ different from s_0

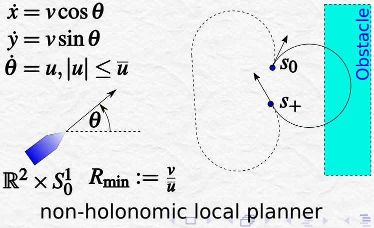
Local planner for a given free space

A device (algorithm) that is fed by a pair of samples (s_0, s_+) and outputs the following:

- Either a path $\mathfrak{P}$ from s_0 to s_+ that goes through the given free space and can be traced by the robot (robots)
- Or a special symbol ✠ that signal about a failure to build $\mathfrak{P}$

The local planner is said to be locally complete if there exists $\varepsilon > 0$ such that the output $\neq$ ✠ whenever the ball with a radius of ε centered at s_0 fully lies in the free space and contains s_+.





Метод случайной дорожной карты

The basic algorithm

The basic algorithm

1. Start with the empty graph

Метод случайной дорожной карты

The basic algorithm

1. Start with the empty graph
2. Pick a probability distribution over the ambient space (simple set covering the free zone)

Метод случайной дорожной карты

The basic algorithm

1. Start with the empty graph
2. Pick a probability distribution over the ambient space (simple set covering the free zone)
3. Repeatedly sample the ambient space according to this distribution; perform a pre-specified number of steps

The basic algorithm

1. Start with the empty graph
2. Pick a probability distribution over the ambient space (simple set covering the free zone)
3. Repeatedly sample the ambient space according to this distribution; perform a pre-specified number of steps
4. Ignore the samples that lie either in an obstacle or outside the working zone. Other samples are is added to the graph

Метод случайной дорожной карты

The basic algorithm

1. Start with the empty graph
2. Pick a probability distribution over the ambient space (simple set covering the free zone)
3. Repeatedly sample the ambient space according to this distribution; perform a pre-specified number of steps
4. Ignore the samples that lie either in an obstacle or outside the working zone. Other samples are is added to the graph
5. When all nodes are build, run over them

Метод случайной дорожной карты

The basic algorithm

1. Start with the empty graph
2. Pick a probability distribution over the ambient space (simple set covering the free zone)
3. Repeatedly sample the ambient space according to this distribution; perform a pre-specified number of steps
4. Ignore the samples that lie either in an obstacle or outside the working zone. Other samples are is added to the graph
5. When all nodes are build, run over them
6. For any node s,

Метод случайной дорожной карты

The basic algorithm

1. Start with the empty graph
2. Pick a probability distribution over the ambient space (simple set covering the free zone)
3. Repeatedly sample the ambient space according to this distribution; perform a pre-specified number of steps
4. Ignore the samples that lie either in an obstacle or outside the working zone. Other samples are is added to the graph
5. When all nodes are build, run over them
6. For any node s,
   - find all its neighbors, i.e., the nodes s_+ with $d(s, s_+) < \varepsilon$, where $\varepsilon > 0$ is a parameter of the algorithm

Метод случайной дорожной карты

The basic algorithm

1. Start with the empty graph
2. Pick a probability distribution over the ambient space (simple set covering the free zone)
3. Repeatedly sample the ambient space according to this distribution; perform a pre-specified number of steps
4. Ignore the samples that lie either in an obstacle or outside the working zone. Other samples are is added to the graph
5. When all nodes are build, run over them
6. For any node s,
   - find all its neighbors, i.e., the nodes s_+ with $d(s, s_+) < \varepsilon$, where $\varepsilon > 0$ is a parameter of the algorithm
   - run over all these neighbors

Метод случайной дорожной карты

The basic algorithm

1. Start with the empty graph
2. Pick a probability distribution over the ambient space (simple set covering the free zone)
3. Repeatedly sample the ambient space according to this distribution; perform a pre-specified number of steps
4. Ignore the samples that lie either in an obstacle or outside the working zone. Other samples are is added to the graph
5. When all nodes are build, run over them
6. For any node s,
   - find all its neighbors, i.e., the nodes s_+ with $d(s, s_+) < \varepsilon$, where $\varepsilon > 0$ is a parameter of the algorithm
   - run over all these neighbors
   - for any of them s_+, apply the local planner to (s, s_+)

Метод случайной дорожной карты

The basic algorithm

1. Start with the empty graph
2. Pick a probability distribution over the ambient space (simple set covering the free zone)
3. Repeatedly sample the ambient space according to this distribution; perform a pre-specified number of steps
4. Ignore the samples that lie either in an obstacle or outside the working zone. Other samples are is added to the graph
5. When all nodes are build, run over them
6. For any node s,
    - find all its neighbors, i.e., the nodes s_+ with $d(s, s_+) < \varepsilon$, where $\varepsilon > 0$ is a parameter of the algorithm
    - run over all these neighbors
    - for any of them s_+, apply the local planner to (s, s_+)
    - draw an edge from s to s_+, if the local planner is able to construct a path

Some ideas of finer sampling near obstacles

- If a sample is within an obstacle, draw a random direction from the uniform distribution, find a free sample in this direction, a finally find the closest free sample in this direction (via e.g., a dichotomy)
- Step aside any new sample according to a Gaussian distribution, only if one of these samples is free and the other is within an obstacle, the free sample from this pair is recorded

Однократное построение маршрута

Find a path between two particular locations ⇒ END

Однократное построение маршрута

Find a path between two particular locations ⇒ END

- Not the entire graph is utilized and needed

Однократное построение маршрута

Find a path between two particular locations ⇒ END

- Not the entire graph is utilized and needed
- Many graph-search methods are incremental, step-by-step; and only a graph-theoretic neighborhood of the current node is processed at every step

Однократное построение маршрута

Find a path between two particular locations ⇒ END

- Not the entire graph is utilized and needed
- Many graph-search methods are incremental, step-by-step; and only a graph-theoretic neighborhood of the current node is processed at every step
- These methods can be run in a situation where the graph is not given prior to the search process, and the graph-theoretic neighborhood of the current node is constructed on-the-fly at every step of the search process as its preliminary stage

Однократное построение маршрута

Find a path between two particular locations ⇒ END

- Not the entire graph is utilized and needed
- Many graph-search methods are incremental, step-by-step; and only a graph-theoretic neighborhood of the current node is processed at every step
- These methods can be run in a situation where the graph is not given prior to the search process, and the graph-theoretic neighborhood of the current node is constructed on-the-fly at every step of the search process as its preliminary stage
- Then the processes of graph searching and graph building go in parallel

Find a path between two particular locations ⇒ END

- Not the entire graph is utilized and needed
- Many graph-search methods are incremental, step-by-step; and only a graph-theoretic neighborhood of the current node is processed at every step
- These methods can be run in a situation where the graph is not given prior to the search process, and the graph-theoretic neighborhood of the current node is constructed on-the-fly at every step of the search process as its preliminary stage
- Then the processes of graph searching and graph building go in parallel
- Building the graph-theoretic neighborhood may be based on data of different kind, e.g., abstract maps, sensory data, data acquired from other centers via communication, etc.

Однократное построение маршрута

Find a path between two particular locations ⇒ END

- Not the entire graph is utilized and needed
- Many graph-search methods are incremental, step-by-step; and only a graph-theoretic neighborhood of the current node is processed at every step
- These methods can be run in a situation where the graph is not given prior to the search process, and the graph-theoretic neighborhood of the current node is constructed on-the-fly at every step of the search process as its preliminary stage
- Then the processes of graph searching and graph building go in parallel
- Building the graph-theoretic neighborhood may be based on data of different kind, e.g., abstract maps, sensory data, data acquired from other centers via communication, etc.
- Using local planner to build neighboring nodes and associated edges

Однократное построение маршрута

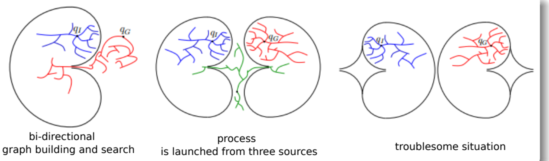
Find a path between two particular locations ⇒ END

- Not the entire graph is utilized and needed
- Many graph-search methods are incremental, step-by-step; and only a graph-theoretic neighborhood of the current node is processed at every step
- These methods can be run in a situation where the graph is not given prior to the search process, and the graph-theoretic neighborhood of the current node is constructed on-the-fly at every step of the search process as its preliminary stage
- Then the processes of graph searching and graph building go in parallel
- Building the graph-theoretic neighborhood may be based on data of different kind, e.g., abstract maps, sensory data, data acquired from other centers via communication, etc.
- Using local planner to build neighboring nodes and associated edges
- The “obstacle-free” path is truncated if it intersects obstacles

Однократное построение маршрута

Find a path between two particular locations ⇒ END

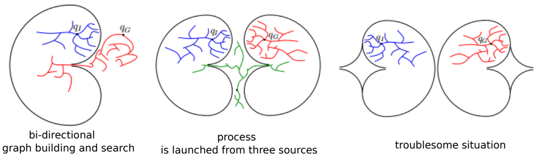
- Not the entire graph is utilized and needed
- Many graph-search methods are incremental, step-by-step; and only a graph-theoretic neighborhood of the current node is processed at every step
- These methods can be run in a situation where the graph is not given prior to the search process, and the graph-theoretic neighborhood of the current node is constructed on-the-fly at every step of the search process as its preliminary stage
- Then the processes of graph searching and graph building go in parallel
- Building the graph-theoretic neighborhood may be based on data of different kind, e.g., abstract maps, sensory data, data acquired from other centers via communication, etc.
- Using local planner to build neighboring nodes and associated edges
- The “obstacle-free” path is truncated if it intersects obstacles

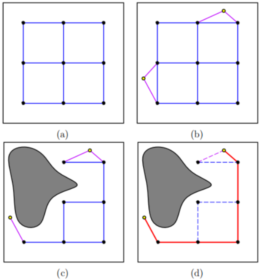


Однократное построение маршрута

Find a path between two particular locations ⇒ END

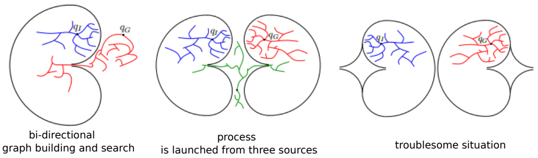
- Not the entire graph is utilized and needed
- Many graph-search methods are incremental, step-by-step; and only a graph-theoretic neighborhood of the current node is processed at every step
- These methods can be run in a situation where the graph is not given prior to the search process, and the graph-theoretic neighborhood of the current node is constructed on-the-fly at every step of the search process as its preliminary stage
- Then the processes of graph searching and graph building go in parallel
- Building the graph-theoretic neighborhood may be based on data of different kind, e.g., abstract maps, sensory data, data acquired from other centers via communication, etc.
- Using local planner to build neighboring nodes and associated edges
- The “obstacle-free” path is truncated if it intersects obstacles

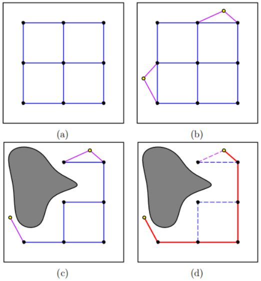




Find a path between two particular locations ⇒ END

- Not the entire graph is utilized and needed
- Many graph-search methods are incremental, step-by-step; and only a graph-theoretic neighborhood of the current node is processed at every step
- These methods can be run in a situation where the graph is not given prior to the search process, and the graph-theoretic neighborhood of the current node is constructed on-the-fly at every step of the search process as its preliminary stage
- Then the processes of graph searching and graph building go in parallel
- Building the graph-theoretic neighborhood may be based on data of different kind, e.g., abstract maps, sensory data, data acquired from other centers via communication, etc.
- Using local planner to build neighboring nodes and associated edges
- The “obstacle-free” path is truncated if it intersects obstacles



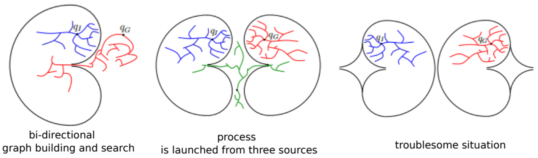


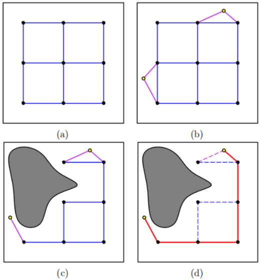
- Multi-step sessions of graph building interspersed with multi-step sessions of graph searching

Однократное построение маршрута

Find a path between two particular locations ⇒ END

- Not the entire graph is utilized and needed
- Many graph-search methods are incremental, step-by-step; and only a graph-theoretic neighborhood of the current node is processed at every step
- These methods can be run in a situation where the graph is not given prior to the search process, and the graph-theoretic neighborhood of the current node is constructed on-the-fly at every step of the search process as its preliminary stage
- Then the processes of graph searching and graph building go in parallel
- Building the graph-theoretic neighborhood may be based on data of different kind, e.g., abstract maps, sensory data, data acquired from other centers via communication, etc.
- Using local planner to build neighboring nodes and associated edges
- The “obstacle-free” path is truncated if it intersects obstacles




(a) (b) (c) (d)

- Multi-step sessions of graph building interspersed with multi-step sessions of graph searching
- Random walks with creating nodes and edges

Typical algorithm

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively

Расширяющиеся деревья

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively
- At each step, a node p is drawn from their union $T := T_{\mathrm{from}} \cup T_{\mathrm{to}}$ according to a probability distribution $P_T(dx)$.

Расширяющиеся деревья

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively
- At each step, a node p is drawn from their union $T := T_{\mathrm{from}} \cup T_{\mathrm{to}}$ according to a probability distribution $P_T(dx)$.
- A probational node p_{prob} is drawn in a neighborhood of p according to an uniform distribution

Расширяющиеся деревья

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively
- At each step, a node p is drawn from their union $T := T_{\text{from}} \cup T_{\text{to}}$ according to a probability distribution $P_T(dx)$.
- A probational node p_{prob} is drawn in a neighborhood of p according to an uniform distribution
- This node is tested to ascertain that it is in the free zone and the local planner can drive the robot from p to p_{prob}. If the test result is positive, p_{prob} and the respective edge are added to the respective graph. Otherwise, p_{prob} is neglected.

Расширяющиеся деревья

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively
- At each step, a node p is drawn from their union $T := T_{\mathrm{from}} \cup T_{\mathrm{to}}$ according to a probability distribution $P_T(dx)$.
- A probational node p_{prob} is drawn in a neighborhood of p according to an uniform distribution
- This node is tested to ascertain that it is in the free zone and the local planner can drive the robot from p to p_{prob}. If the test result is positive, p_{prob} and the respective edge are added to the respective graph. Otherwise, p_{prob} is neglected.
- Periodically, merging of the trees T_{from} and T_{to} is attempted:

Расширяющиеся деревья

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively
- At each step, a node p is drawn from their union $T := T_{\text{from}} \cup T_{\text{to}}$ according to a probability distribution $P_T(dx)$.
- A probational node p_{prob} is drawn in a neighborhood of p according to an uniform distribution
- This node is tested to ascertain that it is in the free zone and the local planner can drive the robot from p to p_{prob}. If the test result is positive, p_{prob} and the respective edge are added to the respective graph. Otherwise, p_{prob} is neglected.
- Periodically, merging of the trees T_{from} and T_{to} is attempted:
  - A node p in one of them is randomly drawn

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively
- At each step, a node p is drawn from their union $T := T_{\mathrm{from}} \cup T_{\mathrm{to}}$ according to a probability distribution $P_T(dx)$.
- A probational node p_{prob} is drawn in a neighborhood of p according to an uniform distribution
- This node is tested to ascertain that it is in the free zone and the local planner can drive the robot from p to p_{prob}. If the test result is positive, p_{prob} and the respective edge are added to the respective graph. Otherwise, p_{prob} is neglected.
- Periodically, merging of the trees T_{from} and T_{to} is attempted:
    - A node p in one of them is randomly drawn
    - Several close nodes of the companion tree are determined

Расширяющиеся деревья

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively
- At each step, a node p is drawn from their union $T := T_{\mathrm{from}} \cup T_{\mathrm{to}}$ according to a probability distribution $P_T(dx)$.
- A probational node p_{prob} is drawn in a neighborhood of p according to an uniform distribution
- This node is tested to ascertain that it is in the free zone and the local planner can drive the robot from p to p_{prob}. If the test result is positive, p_{prob} and the respective edge are added to the respective graph. Otherwise, p_{prob} is neglected.
- Periodically, merging of the trees T_{from} and T_{to} is attempted:
  - A node p in one of them is randomly drawn
  - Several close nodes of the companion tree are determined
  - These nodes are consecutively selected and the possibility to arrive at them from p is examined by using the local planner

Расширяющиеся деревья

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively
- At each step, a node p is drawn from their union $T := T_{\mathrm{from}} \cup T_{\mathrm{to}}$ according to a probability distribution $P_T(dx)$.
- A probational node p_{prob} is drawn in a neighborhood of p according to an uniform distribution
- This node is tested to ascertain that it is in the free zone and the local planner can drive the robot from p to p_{prob}. If the test result is positive, p_{prob} and the respective edge are added to the respective graph. Otherwise, p_{prob} is neglected.
- Periodically, merging of the trees T_{from} and T_{to} is attempted:
  - A node p in one of them is randomly drawn
  - Several close nodes of the companion tree are determined
  - These nodes are consecutively selected and the possibility to arrive at them from p is examined by using the local planner
  - If the positive result of this test is encountered, the nodes are linked with an edge, the graphs T_{from} and T_{to} are viewed as a single whole, and the overall process is terminated

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively
- At each step, a node p is drawn from their union $T := T_{\text{from}} \cup T_{\text{to}}$ according to a probability distribution $P_T(dx)$.
- A probational node p_{prob} is drawn in a neighborhood of p according to an uniform distribution
- This node is tested to ascertain that it is in the free zone and the local planner can drive the robot from p to p_{prob}. If the test result is positive, p_{prob} and the respective edge are added to the respective graph. Otherwise, p_{prob} is neglected.
- Periodically, merging of the trees T_{from} and T_{to} is attempted:
  - A node p in one of them is randomly drawn
  - Several close nodes of the companion tree are determined
  - These nodes are consecutively selected and the possibility to arrive at them from p is examined by using the local planner
  - If the positive result of this test is encountered, the nodes are linked with an edge, the graphs T_{from} and T_{to} are viewed as a single whole, and the overall process is terminated
  - If the search for the positive result fails, the "merging" step is terminated and the stage of expanding T_{from} and T_{to} is resumed

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively
- At each step, a node p is drawn from their union $T := T_{\text{from}} \cup T_{\text{to}}$ according to a probability distribution $P_T(dx)$.
- A probational node p_{prob} is drawn in a neighborhood of p according to an uniform distribution
- This node is tested to ascertain that it is in the free zone and the local planner can drive the robot from p to p_{prob}. If the test result is positive, p_{prob} and the respective edge are added to the respective graph. Otherwise, p_{prob} is neglected.
- Periodically, merging of the trees T_{from} and T_{to} is attempted:
  - A node p in one of them is randomly drawn
  - Several close nodes of the companion tree are determined
  - These nodes are consecutively selected and the possibility to arrive at them from p is examined by using the local planner
  - If the positive result of this test is encountered, the nodes are linked with an edge, the graphs T_{from} and T_{to} are viewed as a single whole, and the overall process is terminated
  - If the search for the positive result fails, the "merging" step is terminated and the stage of expanding T_{from} and T_{to} is resumed

Remark on $P_T(dx)$

- This distribution typically depends on T and varies from step to step.

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively
- At each step, a node p is drawn from their union $T := T_{\mathrm{from}} \cup T_{\mathrm{to}}$ according to a probability distribution $P_T(dx)$.
- A probational node p_{prob} is drawn in a neighborhood of p according to an uniform distribution
- This node is tested to ascertain that it is in the free zone and the local planner can drive the robot from p to p_{prob}. If the test result is positive, p_{prob} and the respective edge are added to the respective graph. Otherwise, p_{prob} is neglected.
- Periodically, merging of the trees T_{from} and T_{to} is attempted:
  - A node p in one of them is randomly drawn
  - Several close nodes of the companion tree are determined
  - These nodes are consecutively selected and the possibility to arrive at them from p is examined by using the local planner
  - If the positive result of this test is encountered, the nodes are linked with an edge, the graphs T_{from} and T_{to} are viewed as a single whole, and the overall process is terminated
  - If the search for the positive result fails, the “merging” step is terminated and the stage of expanding T_{from} and T_{to} is resumed

Remark on $P_T(dx)$

- This distribution typically depends on T and varies from step to step.
- It is designed so that the nodes with lesser number of neighbors be drawn with a higher probability

Typical algorithm

- Two trees T_{from} and T_{to} are built, starting from the initial and destination location, respectively
- At each step, a node p is drawn from their union $T := T_{\mathrm{from}} \cup T_{\mathrm{to}}$ according to a probability distribution $P_T(dx)$.
- A probational node p_{prob} is drawn in a neighborhood of p according to an uniform distribution
- This node is tested to ascertain that it is in the free zone and the local planner can drive the robot from p to p_{prob}. If the test result is positive, p_{prob} and the respective edge are added to the respective graph. Otherwise, p_{prob} is neglected.
- Periodically, merging of the trees T_{from} and T_{to} is attempted:
  - A node p in one of them is randomly drawn
  - Several close nodes of the companion tree are determined
  - These nodes are consecutively selected and the possibility to arrive at them from p is examined by using the local planner
  - If the positive result of this test is encountered, the nodes are linked with an edge, the graphs T_{from} and T_{to} are viewed as a single whole, and the overall process is terminated
  - If the search for the positive result fails, the “merging” step is terminated and the stage of expanding T_{from} and T_{to} is resumed

Remark on $P_T(dx)$

- This distribution typically depends on T and varies from step to step.
- It is designed so that the nodes with lesser number of neighbors be drawn with a higher probability

$N(p)$ — the number of nodes in the vicinity of p, including itself

$$\text{Probability to pick } p := \frac{N(p)^{-1}}{\sum_{\text{all nodes } p'} N(p')^{-1}}$$

Быстро растущее случайное (плотное) дерево

“Rapidly expanding tree” algorithm

Быстро растущее случайное (плотное) дерево

“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \ldots$

Быстро растущее случайное (плотное) дерево

“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \dots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node “source” and no edges

Быстро растущее случайное (плотное) дерево

“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \ldots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node “source” and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed

Быстро растущее случайное (плотное) дерево

“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \dots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node “source” and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed
- The coverage C_k of the graph Γ_k is the union of these paths

Быстро растущее случайное (плотное) дерево

“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \ldots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node “source” and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed
- The coverage C_k of the graph Γ_k is the union of these paths
- At the next step $k+1$, the following is carried out

Быстро растущее случайное (плотное) дерево

“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \dots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node “source” and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed
- The coverage C_k of the graph Γ_k is the union of these paths
- At the next step $k+1$, the following is carried out
  1. The next sample s_{k+1} is drawn from the sequence

Быстро растущее случайное (плотное) дерево

"Rapidly expanding tree" algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \ldots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node "source" and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed
- The coverage C_k of the graph Γ_k is the union of these paths
- At the next step $k+1$, the following is carried out

  1. The next sample s_{k+1} is drawn from the sequence
  2. The "nearest" point $s'_{k+1} \in C_k$ is found

Быстро растущее случайное (плотное) дерево

"Rapidly expanding tree" algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \dots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node "source" and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed
- The coverage C_k of the graph Γ_k is the union of these paths
- At the next step $k+1$, the following is carried out
  1. The next sample s_{k+1} is drawn from the sequence
  2. The "nearest" point $s'_{k+1} \in C_k$ is found
  3. Local planner is used to connect s_{k+1} and s'_{k+1}

Быстро растущее случайное (плотное) дерево

“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \ldots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node “source” and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed
- The coverage C_k of the graph Γ_k is the union of these paths
- At the next step $k+1$, the following is carried out
  1. The next sample s_{k+1} is drawn from the sequence
  2. The “nearest” point $s'_{k+1} \in C_k$ is found
  3. Local planner is used to connect s_{k+1} and s'_{k+1}
  4. If necessary, s_{k+1} is moved along this path towards s'_{k+1} so that the portion between s_{k+1} and s'_{k+1} becomes obstacle-free

Быстро растущее случайное (плотное) дерево

“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \ldots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node “source” and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed
- The coverage C_k of the graph Γ_k is the union of these paths
- At the next step $k+1$, the following is carried out
  1. The next sample s_{k+1} is drawn from the sequence
  2. The “nearest” point $s'_{k+1} \in C_k$ is found
  3. Local planner is used to connect s_{k+1} and s'_{k+1}
  4. If necessary, s_{k+1} is moved along this path towards s'_{k+1} so that the portion between s_{k+1} and s'_{k+1} becomes obstacle-free
  5. Both s_{k+1} and s'_{k+1} are added to the set of nodes of the graph

Быстро растущее случайное (плотное) дерево

“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \ldots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node “source” and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed
- The coverage C_k of the graph Γ_k is the union of these paths
- At the next step $k+1$, the following is carried out

  1. The next sample s_{k+1} is drawn from the sequence
  2. The “nearest” point $s'_{k+1} \in C_k$ is found
  3. Local planner is used to connect s_{k+1} and s'_{k+1}
  4. If necessary, s_{k+1} is moved along this path towards s'_{k+1} so that the portion between s_{k+1} and s'_{k+1} becomes obstacle-free
  5. Both s_{k+1} and s'_{k+1} are added to the set of nodes of the graph
  6. If s'_{k+1} is not a node of Γ_k, every edge of Γ_k that contains s'_{k+1} is divided into two edges

Быстро растущее случайное (плотное) дерево

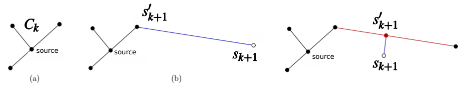
“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \ldots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node “source” and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed
- The coverage C_k of the graph Γ_k is the union of these paths
- At the next step $k+1$, the following is carried out
  1. The next sample s_{k+1} is drawn from the sequence
  2. The “nearest” point $s'_{k+1} \in C_k$ is found
  3. Local planner is used to connect s_{k+1} and s'_{k+1}
  4. If necessary, s_{k+1} is moved along this path towards s'_{k+1} so that the portion between s_{k+1} and s'_{k+1} becomes obstacle-free
  5. Both s_{k+1} and s'_{k+1} are added to the set of nodes of the graph
  6. If s'_{k+1} is not a node of Γ_k, every edge of Γ_k that contains s'_{k+1} is divided into two edges
  7. The path obtained at step 4 is added to the set of the edges

Быстро растущее случайное (плотное) дерево

“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \ldots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node “source” and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed
- The coverage C_k of the graph Γ_k is the union of these paths
- At the next step $k+1$, the following is carried out
  1. The next sample s_{k+1} is drawn from the sequence
  2. The “nearest” point $s'_{k+1} \in C_k$ is found
  3. Local planner is used to connect s_{k+1} and s'_{k+1}
  4. If necessary, s_{k+1} is moved along this path towards s'_{k+1} so that the portion between s_{k+1} and s'_{k+1} becomes obstacle-free
  5. Both s_{k+1} and s'_{k+1} are added to the set of nodes of the graph
  6. If s'_{k+1} is not a node of Γ_k, every edge of Γ_k that contains s'_{k+1} is divided into two edges
  7. The path obtained at step 4 is added to the set of the edges

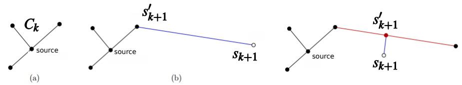


Быстро растущее случайное (плотное) дерево

“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \ldots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node “source” and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed
- The coverage C_k of the graph Γ_k is the union of these paths
- At the next step $k+1$, the following is carried out

1. The next sample s_{k+1} is drawn from the sequence
2. The “nearest” point $s'_{k+1} \in C_k$ is found
3. Local planner is used to connect s_{k+1} and s'_{k+1}
4. If necessary, s_{k+1} is moved along this path towards s'_{k+1} so that the portion between s_{k+1} and s'_{k+1} becomes obstacle-free
5. Both s_{k+1} and s'_{k+1} are added to the set of nodes of the graph
6. If s'_{k+1} is not a node of Γ_k, every edge of Γ_k that contains s'_{k+1} is divided into two edges
7. The path obtained at step 4 is added to the set of the edges



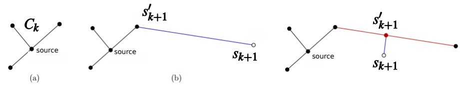
45 iterations

2345 iterations

Быстро растущее случайное (плотное) дерево

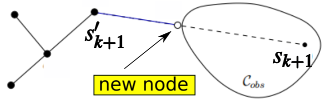
“Rapidly expanding tree” algorithm

- Is based on a machinery to progressively built a random (dense) sequence of samples $s_1, s_2, \ldots$
- At each step k, builds a graph Γ_k with no less than k nodes, starting with the graph with only one node “source” and no edges
- The edges of Γ_k are associated with obstacle-free paths between the samples; to build the paths, a local planner is employed
- The coverage C_k of the graph Γ_k is the union of these paths
- At the next step $k+1$, the following is carried out
    1. The next sample s_{k+1} is drawn from the sequence
    2. The “nearest” point $s'_{k+1} \in C_k$ is found
    3. Local planner is used to connect s_{k+1} and s'_{k+1}
    4. If necessary, s_{k+1} is moved along this path towards s'_{k+1} so that the portion between s_{k+1} and s'_{k+1} becomes obstacle-free
    5. Both s_{k+1} and s'_{k+1} are added to the set of nodes of the graph
    6. If s'_{k+1} is not a node of Γ_k, every edge of Γ_k that contains s'_{k+1} is divided into two edges
    7. The path obtained at step 4 is added to the set of the edges



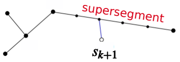
45 iterations

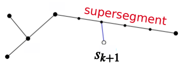
2345 iterations



Некоторые вопросы практической реализации

Некоторые вопросы практической реализации





Kd-tree algorithm

The point of S nearest to s_+

1. Find the median x_* of the projection $\mathrm{Pr}_x(S)$
2. Divide S into two parts: $x \le x_*$ and $x > x_*$
3. Redefine S as the part with the same position w.r.t x_* as s_+
4. Find the median y_* of the projection $\mathrm{Pr}_y(S)$
5. Divide S into two parts: $y \le y_*$ and $y > y_*$
6. Redefine S as the part with the same position w.r.t y_* as s_+
7. go to 1