A mathematical motivation for industrial motion planning

Capco, Jose

23.05.2014

Introduction

Throughout

1. Let $SE(3)$ be the special Euclidean group (group of rigid body transformations) in dimension 3.

2. For any $t \in SE(3)$ let $\mathrm{trans}(t) \in \mathbb{R}^3$ be the translational part of t written canonically as an element in $\mathbb{R}^3$.
3. Let $n \in \mathbb{N}$ be a fixed integer with $n \geq 6$, which we call the *number of joints*.
4. Let C be a subspace of $\mathbb{R}^n$

$$C := \prod_{i=1}^{n} I_i \subset \mathbb{R}^n \qquad I_i = [a_i, b_i] \subset \mathbb{R}, a_i, b_i \in \mathbb{R} \text{ with } a_i \leq 0, b_i \geq 0 \quad \forall i = 1, \ldots, n$$

   which we call the *reachable configuration space* and elements of C are called *(reachable) configurations*.
5. Let π_j be the canonical projection from $\mathbb{R}^n$ to $\mathbb{R}^j$. So we have $\pi_j(C) = \prod_{i=1}^{j} I_i$.
6. Let ρ_j be the canonical projection from $\mathbb{R}^n$ to the j-th coordinate for any $1 \leq j \leq n$. So we have $\rho_j(C) = I_j$.
7. Given a topological space X, by a path we mean a continuous function $p : [0, 1] \to X$.

Introduction

Unless otherwise stated, we fix an abstract *robot* $\mathcal{R}$ which consists of:

- n number of morphisms in the category of smooth manifolds $f_i : \mathbb{R}^i \to SE(3)$ for $i = 1, \ldots, n$. These functions should have the property that for any $1 \leq i < n$ and $c \in C$ one has

$$f_{i+1}(\pi_{i+1}(c)) = f_i(\pi_i(c))g_{i+1}(\rho_{i+1}(c))$$

  where $g_{i+1} : \mathbb{R} \to SE(3)$ is a smooth morphism, we call this the *relative (forward kinematic) function of the i-th link*. We call f_i the *forward kinematic of the i-th link*. And we call f_n simply the *forward kinematic (of the robot)*. We use the symbol F for f_n.

Introduction

- n number of compact subsets of $\mathbb{R}^3$

$$L_i \subset \mathbb{R}^3 \qquad i = 1, \ldots, n$$

We call L_i the *i-th link (of the robot)* and for $c \in C$ we call

$$f_i(\pi_i(c))L_i = \{f_i(\pi_i(c))x \,:\, x \in L_i\} \subset \mathbb{R}^3$$

the *i-th link at configuration c*. If no confusion arises, we sometimes write $L_i(c)$ to mean the i-th link at configuration c (so we can also treat L_i roughly as a function from C to a subset of the power set of $\mathbb{R}^3$ whose elements are all sets which are geometrically congruent to the i-th link).

- A compact (possibly empty) set $K \subset \mathbb{R}^3$ which we call the *(static) collision geometry*.

Definition. We say that there is a *collision at* $c \in C$ iff either one of the following condition holds:

1. $\exists i \in \{1, \ldots, n\}$ such that $L_i(c) \cap K \neq \emptyset$ in this case we also say that the *i-th link collides with the collision geometry at configuration* c.
2. $\exists i, j \in \{1, \ldots, n\}$ with $i \neq j$ and such that $L_i(c)^\circ \cap L_j(c)^\circ \neq \emptyset$ in this case we also say that there is a *self-collsion between links i and j at configuration* c.

If there is a collision at a certain configuration then we say that the *robot collides*, otherwise we say that there is *no collision at* c or that the robot does *not collide*.

Definition. Define

$$C_f := \{c \in C \ : \ \text{there is no collision at } c\}$$

to be the *collision-free (reachable) configuration space* and let $E_f := F(C_f) \subset SE(3)$ be called the *collision-free (reachable) cartesian space*. We can restrict the forward kinematic to $F|_{C_f} : C_f \to E_f$ and if this does not cause confusion we shall also use the symbol F for this restriction.

Introduction

We would like to ”measure” paths in C_f in such a way that we can evaluate whether a path is a good or not. There are many interpretation of a good path in robotics, but in most of these interpretation, one agrees that concatenating two paths is often less desirable and more ”expensive” than just one of the paths, so we will make a definition of a cost function that respects this rule.

Definition. Define $\mathcal{P}$ be the set of paths in C_f and define a binary operator $*$ on $\mathcal{P}$ with the property that

$$p_1 * p_2(t) := \begin{cases} p_1(2t) & t \in [0, \frac{1}{2}] \\ p_2(2t-1) & t \in (\frac{1}{2}, 1] \end{cases}$$

for $p_1(1) = p_2(0)$. Now let $\mathfrak{A}$ be a σ-algebra of a set X, and suppose that $\nu : \mathcal{P} \to \mathfrak{A}$ be a function with the property that $\nu(p_1 * p_2) = \nu(p_1) \cup \nu(p_2)$ if $p_1(1) = p_2(0)$. For an outer measure $\mu : 2^X \to [0, \infty]$ we define $\mu \circ \nu : \mathcal{P} \to [0, \infty]$ to be a *path cost function (induced by ν and μ)*.

Definition. Let E_f° be the interior of E_f in $SE(3)$ and C_f° be the interior of C_f in $\mathbb{R}^n$. Let $q \in C_f^\circ \cap F^{-1}(E_f^\circ)$ and $a := F(q)$, then *there is no kinematic singularity at* q (or sometimes we say there is no kinematic singularity at a, depending on the space we want to focus) iff there exist an open neighbourhood $U_q \subset C_f^\circ$ of q and a chart $\psi : V_a \to V$ of $SE(3)$, with $V_a \subset F(U_q)$ open, $a \in V_a$ and V an open subset of $\mathbb{R}^6$, such that the Jacobian of the function $\psi \circ F|U_q : U_q \to V$ has a rank equal to 6. This is like saying that we can locally apply the implicit function theorem on F at q. Otherwise we say that *there is a kinematic singularity at* q.

Examples

Example. Let $\mathfrak{A}$ be the Borel-algebra of $\mathbb{R}^3$ and let $\nu : \mathcal{P} \to \mathfrak{A}$ be defined by

$$\nu(p) := \{x \in \mathbb{R}^3 \ : \ \exists i = 1, \ldots, n \text{ and } t \in [0,1] \ \nrightarrow \ x \in L_i(p(t))\}$$

then $\nu(p)$ is also called the *swept volume for a path* p. If we take μ to be the Lebesgue measure, then the value of $\nu(p)$ by μ is simply the computed volume of the swept volume in $\mathbb{R}^3$. The cost function from μ and ν is called the swept volume cost function.

Example. Let $\mathfrak{A}$ be the σ-algebra generated by the set of lines in $\mathbb{R}^3$. Let μ be the outer measure on the powerset of $\mathbb{R}^3$ induced from a function that, given a line, evaluates the lines length. Define $\nu : \mathcal{P} \to \mathfrak{A}$ by:

$$\nu(p) := \{(1-t)\text{trans}(F(p(0))) + t\text{trans}(F(p(1))) \ : \ t \in [0,1]\} \subset \mathbb{R}^3 \quad p \in \mathcal{P}$$

so, ν takes a path in the configuration space induces a path in $\mathbb{R}^3$ and takes the line in $\mathbb{R}^3$ with the same endpoint as it.We see that this also satisfies the condition for the definition for the cost function so that $f_c := \mu \circ \nu$ is a path cost function.

Examples

Example. Let $\mathcal{P}_E$ be the set of images of paths in $\mathrm{trans}(E_f) \subset \mathbb{R}^3$ and let $\mathfrak{A}$ be the smallest σ-algebra containing $\mathcal{P}_E$. There is an outer measure induced by the function

$$\mu : \mathcal{P}_E \to [0, \infty]$$

such that if $p : [0, 1] \to \mathrm{trans}(E_f)$ is a path in $\mathrm{trans}(E_f)$ then

$$\mu(p[0, 1]) := \int_0^1 ||J(p(t))||dt$$

where J is the Jacobian of p (so μ just takes the arc-length of p). Define $\nu : \mathcal{P} \to \mathfrak{A}$ by

$$\nu(p) := \mathrm{trans}(F(p[0, 1]))$$

then we see that this satisfies the condition for the definition for the cost function so that $f_c := \mu \circ \nu$ is a path cost function.

Problems

Problem. We state a selection of fundamental problems most common in the area of robotic motion planning:

1. Let $a, b \in E_f$, then find a continuous (not necessarily smooth) path $p \in \mathcal{P}$ such that $F(p(0)) = a$ and $F(p(1)) = b$. This problem is called the *fundamental path planning problem (for points $a, b \in E_f$)* in robotics.
2. Let $a, b \in E_f$ and a cost function $f_c : \mathcal{P} \to [0, \infty]$, then solve the fundamental path planning problem for $a, b \in E_f$ such that $f_c(p)$ is minimum. This problem is a *path optimization problem*.
3. Let $a, b \in E_f$ and a path $p_E : [0, 1] \to E_f$ with $p(0) = a$ and $p(1) = b$, then find a path $p \in \mathcal{P}$ such that $F \circ p = p_E$. This problem is the *path planning problem in the cartesian space for points $a, b \in E_f$*.
4. Let $a, b \in E_f$ then find a continuous path $p \in \mathcal{P}$ such that $F(p(0)) = a$ and $F(p(1)) = b$ such that for all points $r \in [0, 1]$, there is no kinematic singularity at $p(r)$. This is the *path planning problem with (kinematic) singularity avoidance*.
5. Solve the path planning problem with singularity avoidance in such a way that $F \circ p$ is actually a piecewise linear path in E_f.

Path Collision Checks

Definition. A line between a and b in C is called a *direct (robotic) path*. We divide the line between a and b into equal k-parts (using a given metric in $\mathbb{R}^n$) which we call the number of *divisions* for the collision check. The *resolution* is the distance of these equidistant k parts. The *division points* is the set of the endpoints of the k parts, an element of this set is then a *division point*. We say that the direct path between a and b is *discretely colliding* (with the given resolution) if there is a collision at at least one of the division points. Otherwise we say that the direct path between a and b is not discretely colliding. This whole process of testing whether a direct path between point a and b is discretely colliding is called a *direct path discrete collision check* (between points a and b).
A piecewise linear path in C with finite pieces is called a *robotic path*. And a *(robotic) path discrete collision check* is a direct path discrete collision check on all the linear pieces of the robotic path.

The *collision length* of a direct path is just the longest length of a direct path for which there is a collision. More formally:

Definition. Let $L \subset C$ be a direct path of a robot. Then the *collision length* of the path L is

$$\sup\{dist(l) \ : \ l \text{ is a connected component in } L \backslash C_f\}$$

Path Collision Checks

Algorithm: Linear direct path discrete collision check

Input: Division k, configuration points a and b for which a direct path between them should be discretely collision checked

Output: collision true or false

$res = ||b - a||/k$;

v = normalize$(b - a)$;

$t = a - v \cdot res$;

repeat

  $t = t + v \cdot res$;

  if *robot collides at* t **then**

    return true;

  end

until $t = b$;

return false

Path Collision Checks

Construction. Assume first that the number of divisions, k, of a direct path between points a and b is a power of 2. Define C_{-1} to be the empty set and

$$C_i := \{a + (b-a)\frac{j}{2^i} \ : \ j = 0, \ldots, 2^i\} \qquad i = 0, \ldots, \log_2(k)$$

In words C_i is the division points of a direct path between a and b with $2^i \leq k$ as number of divisions. So checking for collision at points in C_i is just doing a direct path collision check with coarser resolution. Algorithm 2 checks for collision with the coarsest division (points a and b) and keeps dividing the number of divisions by two and reiterating the collision check with this number of division, keeping in mind that we need not recheck the division points checked during the coarser path collision check (thus in the loop we check division points in C_i not in C_{i-1}).

Path Collision Checks

Binary direct path discrete collision check when the number of divisions of the direct path is a power of 2.

Algorithm: BinCheck1

Input: Division k that is a power of 2, direct path with endpoints a and b

Output: discrete collision true or false

$j = \log_2(k)$;
$i = 0$;
for $i <= j$ **do**
    if *robot collides at any point in* $C_i \backslash C_{i-1}$ **then**
        return true;
    end
    $i = i + 1$;
end
return false

Path Collision Checks

We can go through the loop by parallel computing and the first positive result will break all the parallel threads/processes.

Algorithm: Binary direct path collision check

Input: Division k , direct path with endpoints a and b

Output: discrete collision true or false

Get the (binary) 2-adic representation of k say:

$k = \sum_{i=0}^{l} 2^{j_i}$;

$res =$ dist(b - a) $/k$;

v = normalize$(b - a)$;

$m = 0$;

for $m < l$ **do**

  if *BinCheck1*$(a + v \cdot res \sum_{i=0}^{m} 2^{j_i}, a + v \cdot res \sum_{i=0}^{m+1} 2^{j_i})$ **then**

    return true;

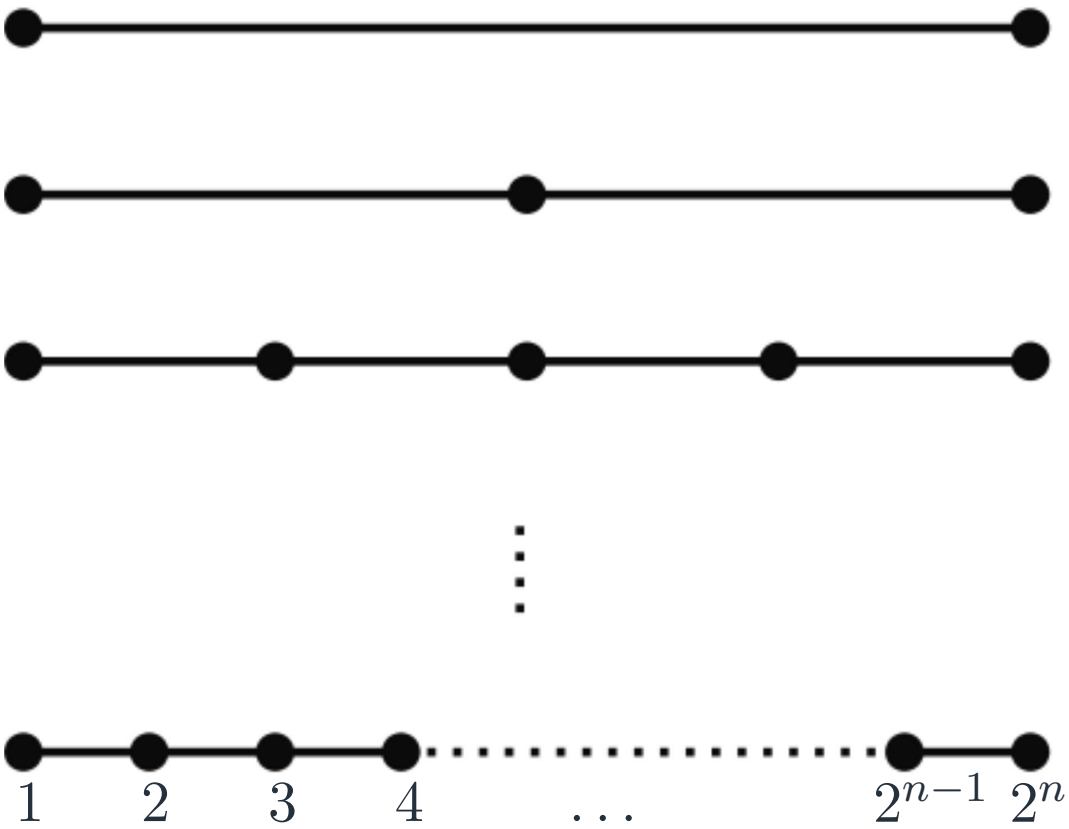
  end

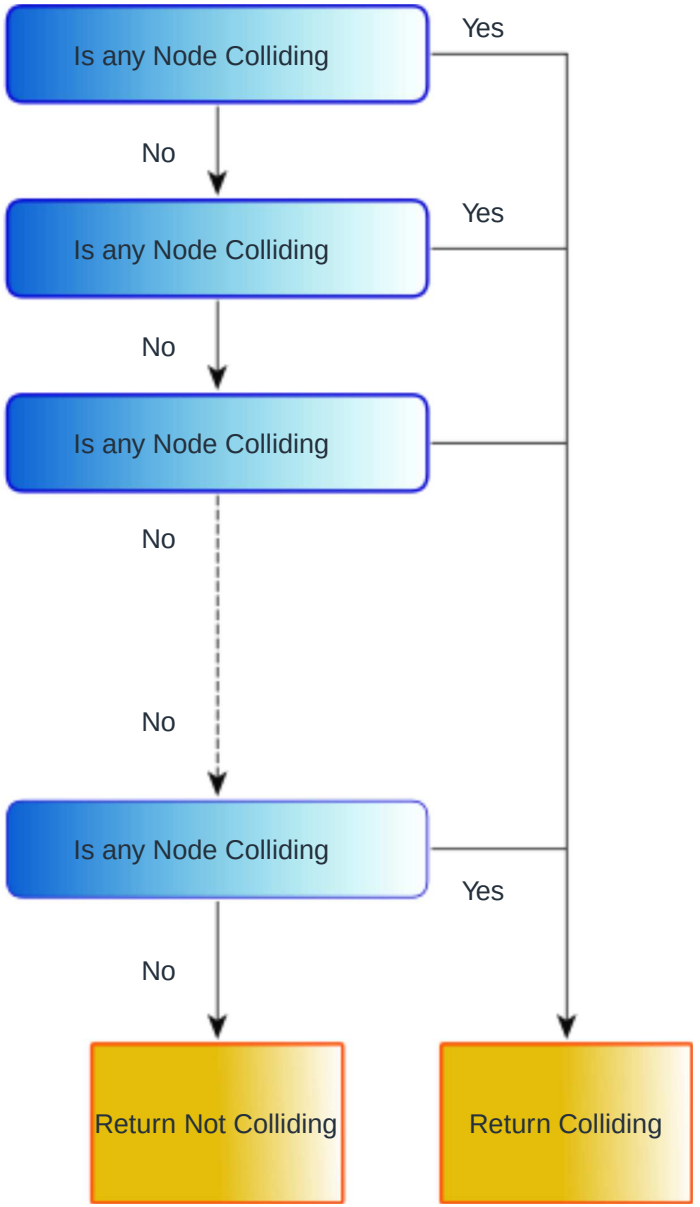
  $m = m + 1$;

end

return false;

Path Collision Checks





RRT Algorithm

Algorithm: Extend

Input: tree T, new node q_{new}, $\epsilon > 0$

Output: new node q_{new}, Advanced or Trapped or Reached

q=randomSample();

$q_{\text{near}} = \text{nearest}(\text{q}, \text{T})$;

$q_{\text{new}} = q_{\text{near}} + \epsilon \cdot \text{normalize}(\text{q} - \text{q}_{\text{near}})$;

if *not* *collide*($q_{\text{near}}, q_{\text{new}}$) **then**

    T.addVertex(q_{new});

    T.addEdge($q_{\text{near}}, q_{\text{new}}$);

    if $q_{\text{new}} = q$ **then**

        return Reached;

    else

        return Advanced;

    end

end

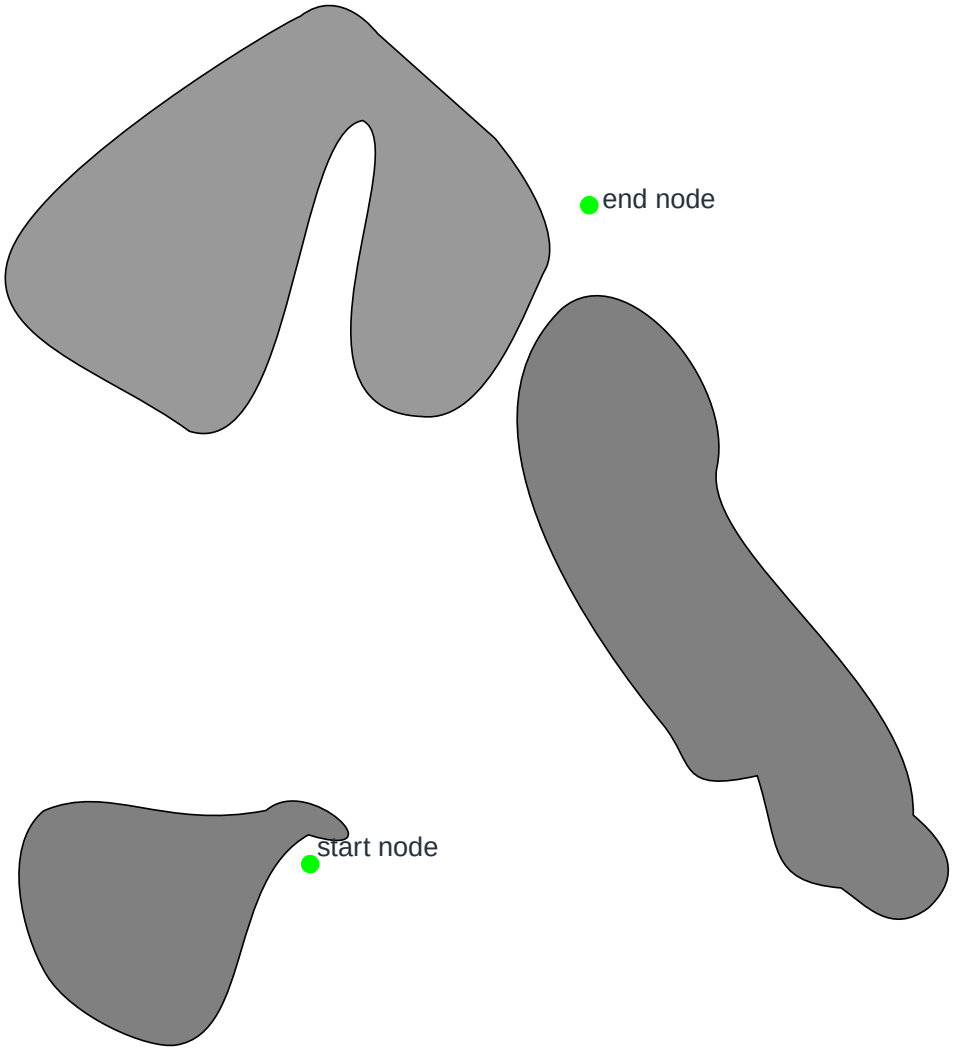
return Trapped;

RRT Algorithm

Algorithm: RRT

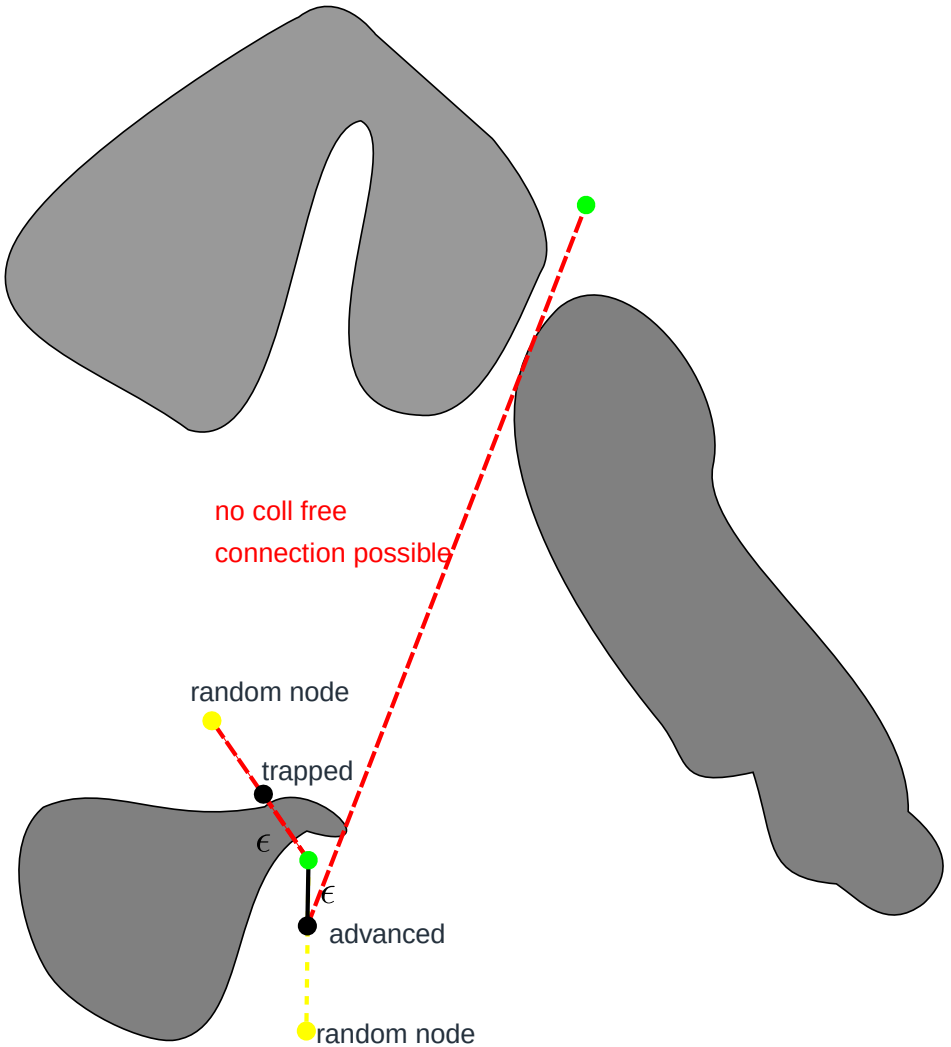
Input: start node q_{start}, goal node q_{goal}, $\epsilon > 0$, timeout
Output: if found within timeout: path from q_{start} to q_{goal}
TreeA = Tree(q_{start});
while *time* $\leq$ *timeout* **do**
    if ***not*** *Extend1(TreeA,* q_{new}, ϵ*)=Trapped* **then**
        if *Connect(*q_{goal}, q_{new}*)=Reached* **then**
            return Path(TreeA, q_{goal});
        end
    end
end

RRT Algorithm



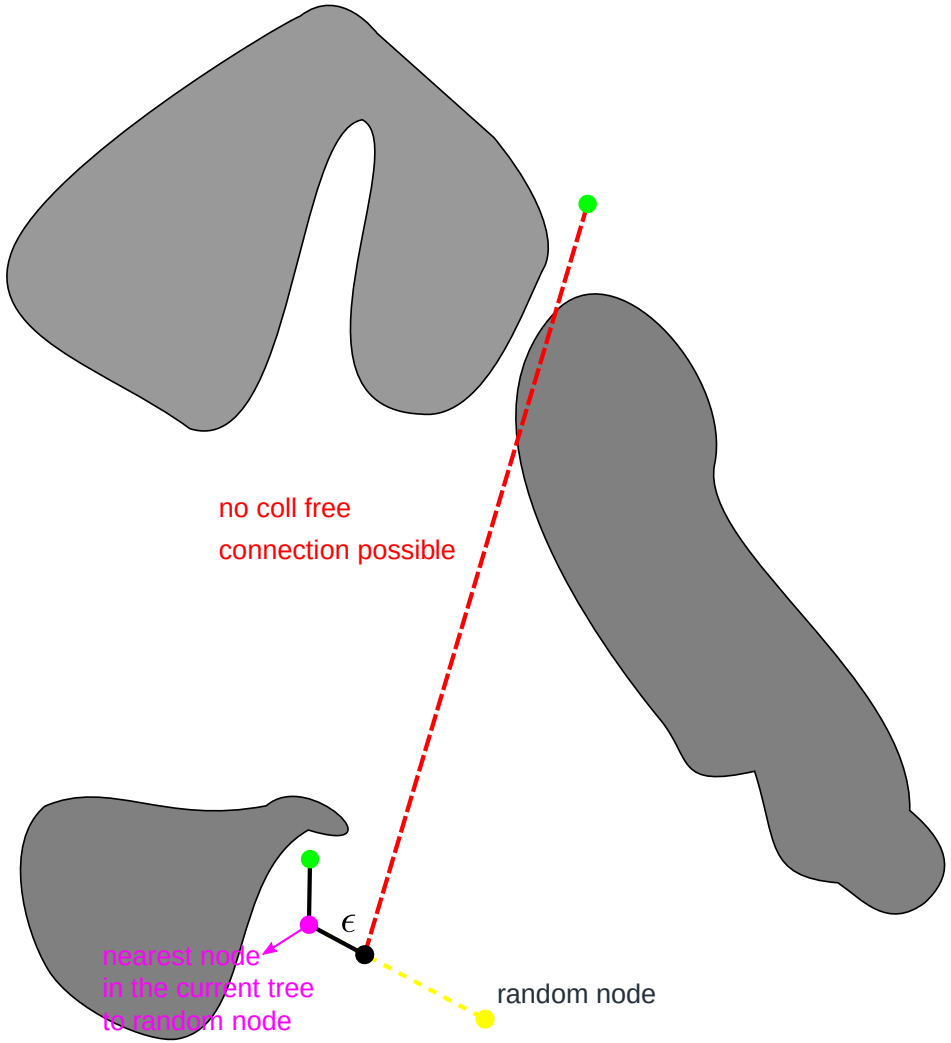
RRT Step 1

RRT Algorithm



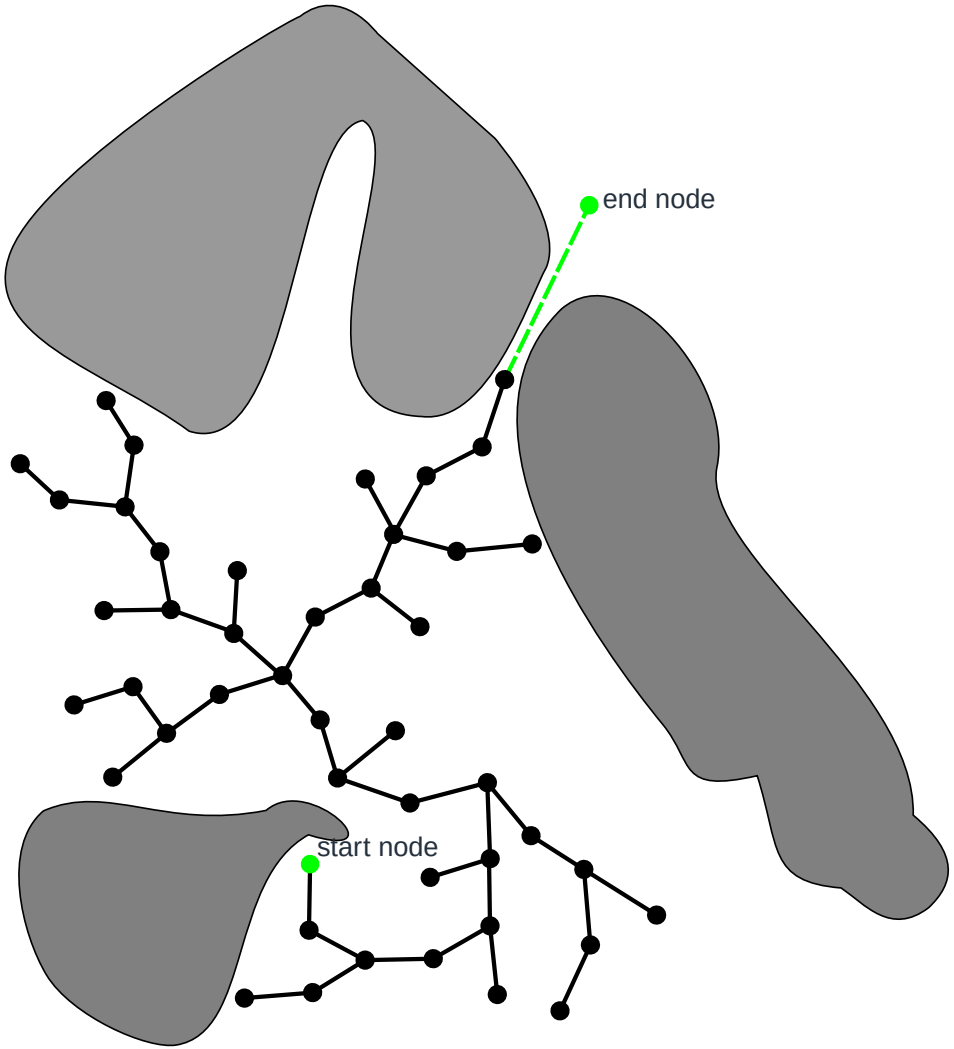
RRT Step 2

RRT Algorithm



RRT Step 3

RRT Algorithm



RRT Step 4

Binpicking

The binpicking problem can be informally and naively described in the following way: Given a robot and a container containing randomly placed workpieces (i.e. congruent compact subsets of $\mathbb{R}^3$), the robot should perform the following actions:

1. Travel collision-free from a start-point to a grip-point of a selected workpiece in the container
2. Grip the workpiece in the container
3. Travel collision-free with the workpiece (thus the last link changes geometry having the workpiece attached to it, this attached workpiece is called a *dynamically attachable frame* or DAF) to a certain goal-point and release the workpiece at the goal-point
4. Repeat the last three steps until either the container is empty or a collision-free path by the latter steps are not possible anymore.

Binpicking

Definition. Given an abstract robot $\mathcal{R}$, a *robot based on* $\mathcal{R}$ is another robot $\mathcal{R}'$ with everything similary defined as $\mathcal{R}$ except that the n-th link L'_n is possibly another compact subset of $\mathbb{R}^3$ that contains L_n and the static collision geometry K' is possibly different from K. Thus the collision-free configuration and cartesian spaces C'_f and E'_f respectively are different.

Now we are ready to formally define the binpicking problem:

Definition. Let $\mathcal{R}$ be an abstract robot. Let $\mathcal{R}_1, \ldots, \mathcal{R}_k$ with $k \in 2\mathbb{N}$ be abstract robots based on $\mathcal{R}$ with collision-free configuration spaces $C_1, \ldots, C_k$ and collision-free cartesian spaces $E_1, \ldots, E_k$. Now let

$$\{(a_i, b_i) \in E_i^2 \ : \ i = 1, \ldots, k\}$$

Then solve the fundamental path planning problems for robots $\mathcal{R}_i$ with endpoints a_i and b_i.

Coverage Planning

Definition. Given a finite sequence of k points in E_f (with $k \geq 2$) say

$$S_k := \{e_1, e_2, \ldots, e_k\} \subset E_f$$

and a path cost function $f_c : \mathcal{P} \to \mathbb{R}^+$. Find a permutation (re-ordering) $\sigma : S_k \to S_k$ such that

1. There are speicific solutions

   $$\{p_1, \ldots, p_{k-1}\} \subset \mathcal{P}$$

   to the fundamental path-planning problems for the pairs $(e_{\sigma(i)}, e_{\sigma(i+1)})$ for $i = 1, \ldots, k-1$.
2. For any other permutation of $\tau : S_k \to S_k$ and solutions

   $$\{p'_1, \ldots, p'_{k-1}\} \subset \mathcal{P}$$

   of the fundamental path-planning problems for the pairs $(e_{\tau(i)}, e_{\tau(i+1)})$ for $i = 1, \ldots, k-1$, we have

   $$\sum_{i=0}^{k-1} f_c(p_i) \leq \sum_{i=0}^{k-1} f_c(p'_i)$$

The solution to this problem (i.e. solving the permutation σ and the paths $\{p_1, \ldots, p_k\}$ is the solution to the *coverage planning problem* (for k-points in the collision-free reachable cartesian space).

References

[1] **C. Ericson**,
Real-Time Collision Detection, Morgan Kaufmann Publishers 2005

[2] **L-P. Ellekilde, J.A. Jorgensen**,
RobWork: A Flexible Toolbox for Robotics Research and Education, International Symposium on Robotics (ISR) 2010, Munich, Germany

[3] **S. Karaman, E. Frazzoli**,
Incremental Sampling-based Algorithms for Optimal Motion Planning, International Journal of Robotics Research, Vol. 30, No. 7, June 2011

[4] **L-E. Kavraki, J-C. Latombe, M.H. Overmars, P. Svestka**,
Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces, IEEE Transactions on Robotics and Automation, Vol. 12, pages 566-580, 1996

[5] **Kineo C.A.M.**,
KineoWorks^{TM},
http://www.kineocam.com/kineoworks-library.php
Last Accessed: 23.01.2013

[6] **J.J. Kuffner, S.M. LaValle**,
RRT-Connect: An Efficient Approach to Single-Query Path Planning, IEEE International Conference on Robotics and Automation 2000

[7] **S.M. LaValle**,
Planning Algorithms, Cambridge University Press, Copyright S.M. Lavalle 2006, http://planning.cs.uiuc.edu/

[8] **S.M. LaValle, J.J. Kuffner**,
Randomized Kinodynamic Planning, International Journal of Robotics Research, Vol. 20, No. 5, pp. 378-400, May 2001

[1] **M. Strandberg**,
Yet Another OBB-Tree Implementation, http://sourceforge.net/projects/yaobi/, 2005
Last Accessed: 01.06.2011